RL-TR-97-35
Final Technical Report
June 1997

THE COMMON PROTOTYPING ENVIRONMENT

BBN Systems and Technology

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 7703

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-97-35 has been reviewed and is approved for publication.

APPROVED:

DONALD F. ROBERTS
Project Engineer

FOR THE COMMANDER:

JOHN A. GRANIERO, Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL/C3CA, 525 Brooks Rd, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

THE COMMON PROTOTYPING ENVIRONMENT

Contractor: BBN Systems & Technology
Contract Number: F30602-91-C-0014
Effective Date of Contract: 5 December 1990
Contract Expiration Date: 28 February 1996
Program Code Number: 3E20
Short Title of Work: The Common Prototyping Environment

Period of Work Covered: Dec 90 - Feb 96

Principal Investigator: Mark H. Burstein
Phone: (617) 873-3861

RL Project Engineer: Donald F. Roberts
Phone: (315) 330-3577

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by Donald F. Roberts, RL/C3CA, 525 Brooks Rd, Rome, NY.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED	
	June 1997	Final	Dec 90 - Feb 96

4. TITLE AND SUBTITLE

THE COMMON PROTOTYPING ENVIRONMENT

5. FUNDING NUMBERS

C - F306702-91-C-0014
PE - 63728F
PR - G703
TA - 00
WU - 01

6. AUTHOR(S)

Mark H. Burstein

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

BBN Systems and Technology
10 Moulton Street
Cambridge MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER

BBN Report No. 8151

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

Rome Laboratory/C3CA
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-97-35

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Donald F. Roberts/C3CA/(315) 330-3577

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release, distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Common Prototyping Environment Abstract: The ARPA/Rome Laboratory Knowledge Based Planning and Scheduling Initiative (ARPI) is an Advanced Research Projects Agency (ARPA) and Rome Laboratory (RL) sponsored initiative to promote the development of new knowledge-based planning and scheduling technology for use by the military in support of operational planning and scheduling problems. The ARPI Common Prototyping Environment Project was established in order to facilitate the communication of an understanding of military problems to the research community, provide sample data and problems, promote the sharing and reuse of software tools, and ultimately provide a platform of the experimental testing and integration of research products, so that successful technologies could be demonstrated solving operational problems. The CPE team and BBN Systems and Technologies and ISX Corporation was tasked to provide a wide variety of infrastructure services for the initiative, including knowledge representation tools, and assistance in developing integration experiments and software feasibility demonstrations (IFDs).

14. SUBJECT TERMS

Knowledge Based Planning, Scheduling, Transportation Scheduling, Prototyping

15. NUMBER OF PAGES

102

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

APPENDICES

LIST OF FIGURES

1.0 INTRODUCTION

The ARPA/Rome Laboratory Knowledge Based Planning and Scheduling Initiative (ARPI) is an Advanced Research Projects Agency (ARPA) and Rome Laboratory (RL) sponsored initiative to promote the development of new knowledge-based planning and scheduling technology for use by the military in support of operational planning and scheduling problems. Included in the program are research efforts aimed at developing new AI planning technologies in the areas of plan generation and replanning, schedule optimization, formal plan validation, decision-theoretic planning, heterogeneous database access, reasoning under uncertainty, and case-based reasoning. An important goal of the ARPI program is to foster the integration of these technologies, and transition successful applications of these technologies to operational prototypes that can effectively demonstrate their utility to the military.

The Common Prototyping Environment Project was established in order to facilitate the communication of an understanding of military problems to the research community, provide sample data and problems, promote the sharing and reuse of software tools, and ultimately provide a platform for the experimental testing and integration of research products, so that successful technologies could be demonstrated solving operational problems (See Figure 1). The CPE team of BBN Systems and Technologies and ISX Corporation was tasked to provide a wide variety of infrastructure services for the initiative, including the collection and dissemination of domain data and reference materials, support software, knowledge representation tools, and services including the organization of annual ARPI workshops and quarterly meetings, assistance in developing integration experiments and software feasibility demonstrations (IFDs). They developed the CPE Testbed as a distributed platform for researchers to build and conduct collaborative experiments and demonstrations.

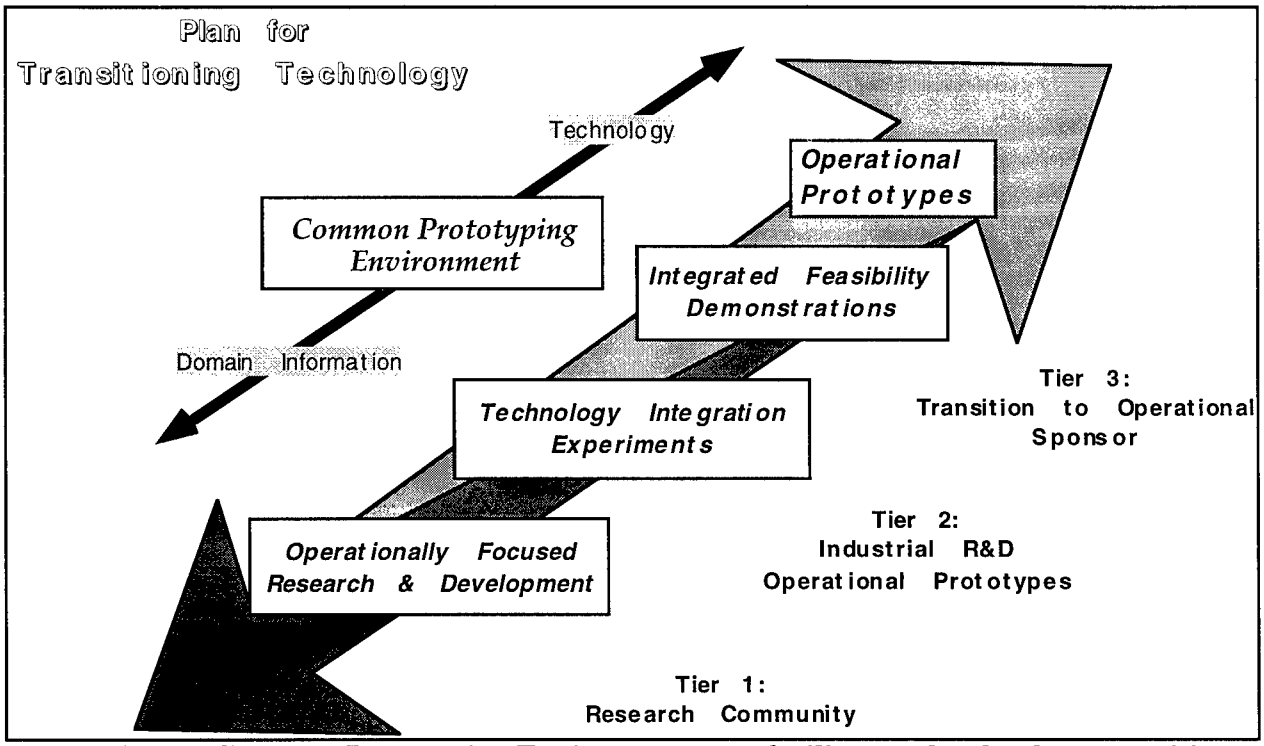

Figure 1: The Common Prototyping Environment as a facilitator of technology transition.

The general organization of the initiative is based on three tiers of efforts (See Figure 2). *Tier 1* involves a number of independent research efforts, primarily at universities, that are oriented

toward developing fundamental knowledge-based reasoning capabilities which can be applied to operational planning and scheduling problems. *Tier 2* efforts, taking place primarily at industrial research and development laboratories, are aimed at taking these fundamental technological capabilities and producing *Technology Integration Experiments* (TIEs) that are attempts to merge many of the individual developments in Tier 1 into coordinated systems that demonstrate the application of the Tier 1-developed technologies to an operational problem. *Tier 3* involves technology transfer of knowledge-based planning technology via the development of *Integrated Feasibility Demonstrations* (IFDs) and user-supported operational prototypes. Each Tier 2 laboratory was made responsible for the development and hardening of research at Tier 1 into software that could be demonstrated in an operational domain.

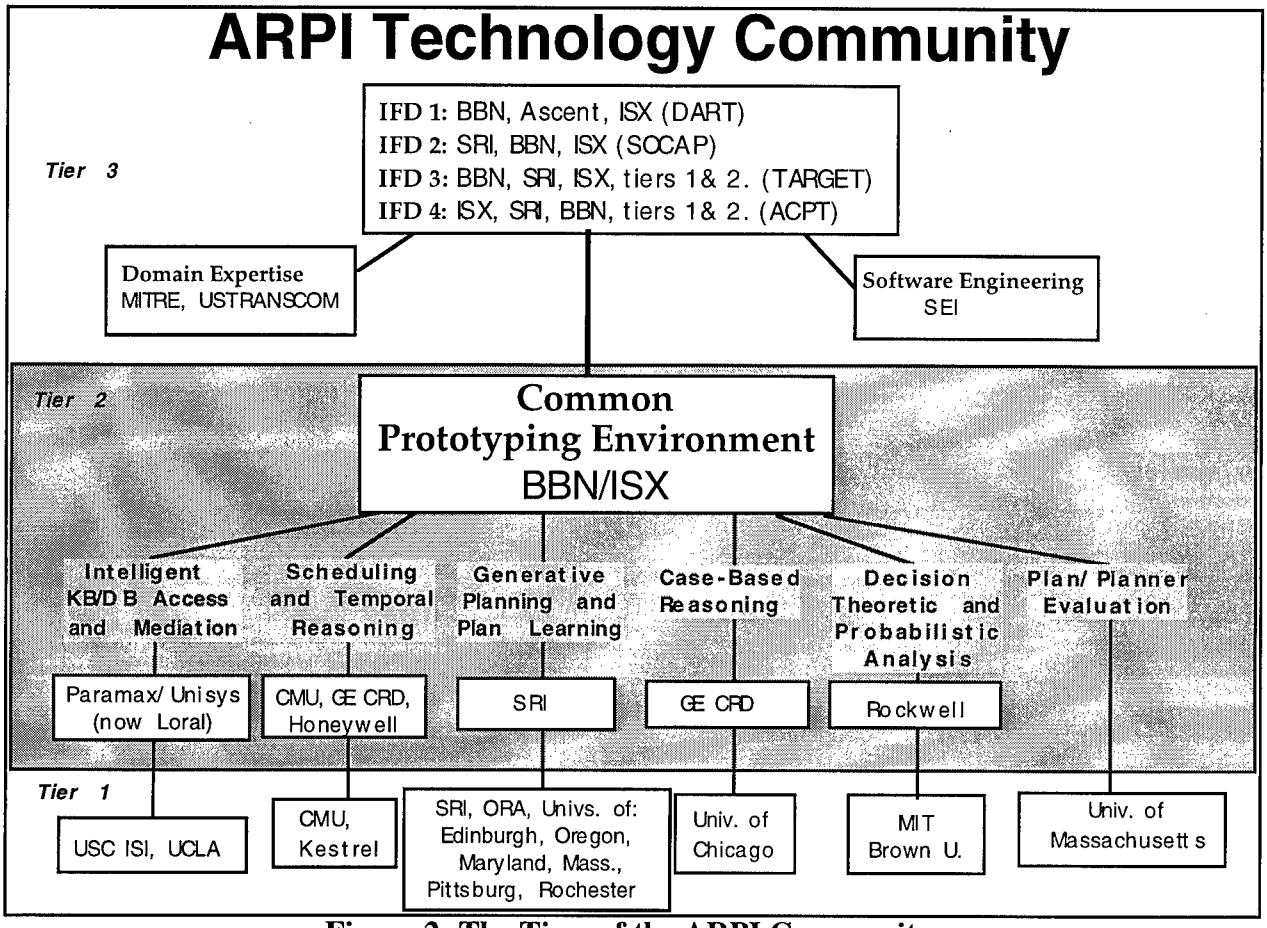

Figure 2: The Tiers of the ARPI Community

From an operational perspective, the ARPI focused for the first three or four years on the problem of *MilitaryTransportation Planning*[1]. The initial *Tier 3* effort (for IFD-1) was the development of a transportation planning tool called DART (Dynamic Analysis and Replanning Tool). DART provided transportation planners with an ability to rapidly generate, evaluate and modify transportation plans. An initial version of DART was deployed to support transportation planning during the DESERT SHIELD/DESERT STORM operation. The second IFD demonstrated SOCAP, a plan generation system developed as an application of the SIPE-2 plan

1. See (Burstein, 1991), BBN Technical Report 7495, for a brief description of the military transportation planning domain, and an early description of the CPE.

generation system by SRI International. The third IFD demonstrated TARGET, a collaborative planning tool that provided an object-oriented plan database and interface, while providing an integrating platform for demonstrating some of the case-based reasoning, decision-theoretic analysis, constraint-based scheduling and knowledge/data representation techniques that had been developed. The fourth IFD demonstrated these and additional technologies applied in a tool for Air Campaign Planning (ACPT).

In addition to the individual research and development efforts, the ARPI organized a number of issue and technology working groups. The issue working groups were concerned with the general management and scheduling of the Tier 1 and Tier 2 activities. The four working groups discussed

- The Technology Roadmap
- Knowledge Representation, Knowledge Acquisition and System Architecture Issues
- The Common Prototyping Environment
- IFDs and a "Visionary Demonstration"

In order to guarantee that research developments are focused on operational needs, an issue working group was established to maintain a *technology roadmap.* The Technical Roadmap provides a general plan/schedule for the developments to occur on the Tier 1 efforts, and their impact on Tiers 2. An early version of this roadmap is shown in Figure 3. Overall, the Technical Roadmap was expected to:

- identify specific technology developments to occur as part of the ARPI and included a schedule that anticipated when these technology developments would be available.
- identify the operationally-relevant functional capabilities that each technology development would provide.
- identify a series of *Critical Experiments* that would serve to evaluate the maturity of the individual technology developments. Specifically, it included a schedule for the IFDs, and the technical/functional capabilities that will be demonstrated at each IFD.

The Knowledge Representation, Acquisition and System Architecture group was responsible for developing representational structures, requirements for domain knowledge, and architectures and functional description for planning system components that will promote collaborative work and knowledge sharing among members of the PI's research community. This includes the development and dissemination of conventions and constraints on representational structures, reasoning processes, and the architectural infrastructure. Their primary product was a specification of the KRSL language for describing shared representations of content.

The Feasibility and Visionary Demonstrations working group was responsible for characterizing a future system for Joint Operations Planning that reflected the application of new technologies developed by DARPA, Rome Laboratory and other organizations. The specific focus was on technologies to be developed as part of the ARPI. The Visionary Demonstration that was ultimately developed was in the form hypermedia presentation depicting a technology-supported Joint Operational Planning scenario. Specific aspects of the Visionary Demonstration were implemented in the IFDs.

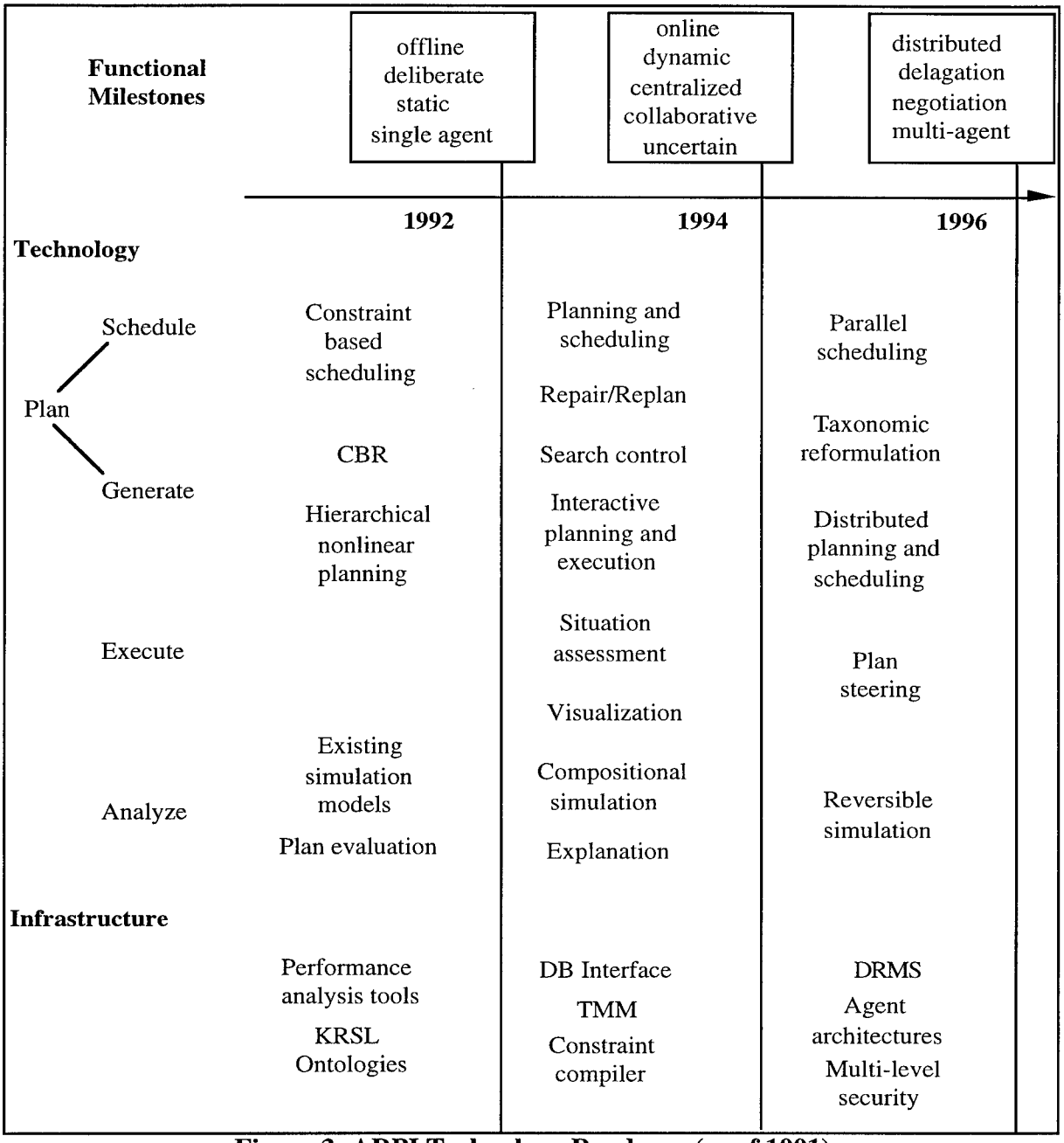

Figure 3: ARPI Technology Roadmap (as of 1991)

The Common Prototyping Environment (CPE) working group was responsible for specifying requirements for the Common Prototyping Environment such that it would provide both developmental support and a testbed for accumulating research results and assessing research progress.

2.0 THE COMMON PROTOTYPING ENVIRONMENT

The Initiative involves dozens of researchers at many sites, each working on basic or applied research with a view to supporting large scale, spatially- and functionally-distributed planning and scheduling systems for the deployment and employment of military resources. The researchers in these groups are working on different aspects of the overall problem, and in many

cases, have different goals and criteria for the success of their research programs. For example, while one group was working on the adaptation of existing generative planning technology to a near-term demonstration of automated planning in the transportation domain, another was attempting to formalize concepts such as "cooperation" between automated agents in a distributed planning system. At the same time, a third group was developing scheduling algorithms that are tractable at the large scales inherent in the military transportation planning problems routinely faced by the Joint Staff.

In support of the initiative, a Common Prototyping Environment (CPE) was envisioned to alleviate the tremendous burden all of these researchers would otherwise bear in individually acquiring data and knowledge of the domain, and in coordinating the development of their software systems. The general purpose of the common prototyping environment (CPE) was to encourage convergence between all of the researchers taking part in the ARPI. The ultimate goal was a complete, operational transportation planning system to which all participants of the ARPI contributed technology. Along the way to this ultimate goal, the CPE would serve to support cooperative software development, and provide an environment within which all participants can run experiments and derive scientific results on the effectiveness of their individual technologies.

BBN Systems and Technologies and ISX Corporation were teamed to provide the CPE, supporting the research community by providing access to realistic military planning problems, while reducing each individual project's overhead by providing a suite of sharable software tools that would foster the integration, evaluation and transitioning of the planning and scheduling technologies developed. The CPE development initially supported the community through the **CPE Repository**, an "electronic clearinghouse" for software (sources, executables, and sometimes both) and representative data and knowledge, in both textual and electronic forms, about the military transportation planning process. The **CPE Testbed** was developed in 1993-4, as a distributed environment for continuing the series of **Technology Integration Experiments** (TIEs). The CPE Testbed, consists of a suite of software tools that enable systems implementing specific technological techniques to be experimentally integrated in a distributed knowledge-based communications environment.

The initially stated goals for the CPE were:

- To provide software, data and domain representations as communal resources on a standard hardware platform for research and development in the domain of military transportation planning and scheduling.
- To form a repository for examples, test cases, open problems and fruitful approaches to solving those problems, or supporting technologies that might be used in developing such solutions.
- To provide an environment in which solutions to similar problems can be compared and contrasted on an even footing.
- To facilitate transfer of data and problems to the research community and potential solutions to operational technology developers.
- To avoid unnecessary "reinventing of wheels" for interfaces, data access tools, basic domain representations.

From the outset, BBN and ISX sought to fulfill these objectives in a number of different ways. ISX took primary responsibility for the collection and dissemination of domain data and information, to be provided in the form of "Domain Packages", the first being for the joint military transportation planning domain. At the same time BBN collected and organized a set of primarily LISP-based software tools to form the core of an "ARPI community standard" hardware/software environment. BBN and ISX worked together, with input from the ARPI

community, to develop a Knowledge Representation Specification Language (KRSL) that could be used as a common language for representing domain information and communicating between the different AI-based software technologies developed under the initiative. KRSL was built on top of the LOOM knowledge representation system from USC Information Sciences Institute.

Throughout the course of the initiative, the primary means of disseminating the tools and data collected for the CPE was through the **CPE Repository**, an internet-accessible (by FTP) repository housed at Rome Laboratory. Later, as described below, a distributed **CPE Testbed** was also developed, enabling ARPI participants to work with software systems running at remote sites, and develop experiments and demonstrations where their technologies were used in conjunction with those remote systems.

The CPE Testbed grew out of experiences in collaborative projects from the first several years of the Planning Initiative. These included our participation in the cooperative development of software demonstrations called **Integrated Feasibility Demonstrations** (IFDs) (see accompanying article) and smaller-scale, **Technology Integration Experiments** (TIEs) where different research and development teams explored the issues involved in successfully combining their research software and techniques.

The CPE Testbed was first released in May of 1993. It capitalized on all of the TIEs conducted during 1992, bringing the software used in all of those TIEs into a common distributed environment and using a single knowledge representation language for information exchange (KRSL). The CPE Testbed was also designed to support experiments in which components were combined in novel ways, or directly compared doing similar functions. To address the issue of experimentation, the testbed included a suite of tools for collecting and analyzing data about the performance of the modules used in solving planning problems. The CPE Testbed ultimately included technology components developed in more than ten different R&D laboratories sponsored under the initiative. These software systems all provide services that can be utilized in experiments based on solving military planning problems, even when the software systems are physically hosted at sites dispersed throughout the country.

The CPE Repository

As originally envisioned, the Common Prototyping Environment was to be built up as a layered set of reusable software tools, some to be purchased commercially, others to be provided through the CPE Repository as contributions from ARPI participants. A number of software components used in the DART system were provided for use by other initiative members as part of the CPE. Domain Package information, especially the useful data that had been collected about the transportation planning domain. This information was "cleaned up" and provided in a form that could be more easily incorporated into research projects as background for demonstrations and research development efforts alike. Overall, The layers of the environment were as follows:

Planning Support tools:

- Temporal Reasoning Systems
- Uncertainty Maintenance Systems
- Case indexing and Retrieval Mechanisms
- Generative Planning Tools

Modeling Support

- Scenario Description and Aggregate Force Model Editing Tools

Simulation and Analysis Tools

- Domain Simulations and associated graphics
- Transportation Analysis Tools
- Decision Support Tools

Domain Information

- Sample Domain Data in ASCII and RDBMS formats
- Object descriptions in LOOM and KRSL
- Sample problems, examples, and other supporting textual materials

Prototyping Environment Support

- System Performance Metering Tools
- Reusable CLIM-based Interface tools provided by BBN and others.

Baseline System (Mostly COTS)

- Knowledge Representation tools (LOOM, KRSL)
- SQL DBMS (Oracle) and LISP-SQL interface
- Common Lisp Interface Manager (CLIM) for developing GUIs.

Hardware/Software Substrate (COTS)

- X Windows
- Common Lisp, other programming languages as needed.
- Ethernet, TCP/IP
- UNIX OS
- Sun Workstations

Throughout the course of the initiative, demonstration systems from each ARPI contractor, technology component systems (such as temporal reasoners, decision support tools, planners, schedulers), and domain-specific software systems developed for IFDs and TIEs were added to the repository. ISX also collected an extremely large amount of textual materials and databases that were used in planning and executing the various IFDs, and added this information to the repository. Where appropriate, KRSL forms of the data were also made available. The Electronic Card Catalog for the ARPI CPE Repository contains well over 300 items at the present time.

The CPE Testbed

For the first IFDs and TIEs, ARPI contractors developed and used largely domain-independent AI tools to address military planning problems. After the initial round of TIEs was completed, all of the technology components involved in the TIEs were integrated into the CPE Testbed, a single environment based on the interoperability model used in the "Infrastructure TIE", which explored the use of KQML, the Knowledge Query Meta Language, as a message passing model for distributed knowledge-based communications, and introduced Paramax' (now Loral) LIM/IDI mechanism for connecting KRSL/LOOM representations to SQL databases. The CPE Testbed was designed to:

- Provide an infrastructure for future TIEs that supported experimental metrics and their evaluation and uniform access to data and scenarios for the transportation planning domain.

- Establish a common representational basis for all remote communications between planning and scheduling systems.
- Facilitate the transition of technologies into IFDs and operational prototypes.
- Enable the exploration of coordination and control issues in a heterogeneous system for replanning and rescheduling.

CPE Testbed Release 1.0 was developed in an intense five month effort during the spring of 1993 involving all of the participants in TIEs. During that time, all of the ad-hoc communications that has occurred in the TIEs were transformed into consistent KRSL representations. CPE Release 2.0, which included a number of additional modules and improved experimental analysis support was released in the February of 1994.

The CPE Testbed was designed to run experiments that tested the speed and effectiveness of new technologies, both singly and in TIE-like combinations. To run an experiment in the CPE, one first chooses sets of modules that will run in each of a number of 'trials'. One defines the data to be collected using a mechanism for adding 'alligator clips' or **meters** at various points in the computations of modules, and in their communications with other modules.

The CPE successfully demonstrated the potential for increased collaboration and sharing among a large set of research and development projects. It served as a validation of the concept and methodology for promoting more effective software and information sharing, moved a large community of researchers toward mechanisms for validating the products of their work both in terms of increased effort to demonstrate interoperability and increased attention to the establishment of experimental metrics for progress. The CPE Testbed facilitated a number of experiments in which similar technologies were compared, and compatible technologies were integrated to exhibit improved planning capabilities. The CPE Testbed is described in detail in the paper in Appendix B, and in the CPE Testbed Users Manual and Reference Guide.

3.0 A BRIEF HISTORY OF THE CPE PROJECT

In the Beginning - DART

To a large extent planning for the ARPI program began with a DARPA (now ARPA)-sponsored planning workshop in December of 1989. At this workshop issues relating to the potential application of AI planning technology to transportation scheduling and planning were examined. One result of this workshop was the identification of fundamental research issues that needed to be addressed in order to promote the application of AI planning technology to operational crises action planning problems. As a result of the workshop, a BAA was released in April 1990. This BAA solicited Tier 1 and Tier 2 proposals.

The ARPI program was initiated in January of 1990. Work by BBN, ISX, and Ascent Technologies on the DART demonstration system began shortly thereafter, along with other scheduling technology demonstrations by MITRE (MDART) and CMU (CDART). The objective of the DART project was to provide a quick demonstration of the operational impact of knowledge-based planning and scheduling technology on transportation planning at USTRANSCOM (US Transportation Command). Installation of basic support software at USTRANSCOM was completed in May 1990. Demonstrations of DART's operational concepts were held July 25 and August 3, and initial installation of DART software at USTRANSCOM J5 occurred August 13-17. Shortly after that the DART project was redirected to support Desert Shield. After a period of intense development that fall and winter, DART was successfully used to assist transportation planning for portions of this operation. The DART workstation environment which was installed at USTRANSCOM at that time was credited with reducing routine plan analysis from 3 days to 1 day.

In addition to providing an early standard-setting demonstration of the state of the art in planning technology, the DART effort set standards and expectations for the application of knowledge-based planning and scheduling technology, promoting necessary changes in the operational domain for introducing new technologies, and exploring the impact and effectiveness of strategies for focusing related research projects and transitioning new technology to operational domains.

During the November and December of 1990, BBN also teamed with CMU to turn the core of CMU's CDART demonstration into a research tool called the Prototype Feasibility Estimator (PFE). PFE was a LISP-based transportation simulation model that produced results very similar to the RAPIDSIM model used in the deployed DART system. PFE later served as an important domain model for much of the constraint-based scheduling work done under the auspices of the initiative. PFE is described in detail in (Burstein & Kosy, 1991)

First Year Activities - ARPI Kickoff, CPE Repository and IFD-2

The kickoff meeting for the ARPI, organized by BBN and ISX, occurred in February 1991. Initial plans for the Common Prototyping Environment were distributed and discussed. During the meeting, several working groups were established to support management and coordination of the Tier 1 research and to plan the sequence of IFDs. (Tiers 2, 3). After several meetings a set of issue working group reports were generated and compiled into a single informal document. This document was distributed in October 1991. In November 1991 a workshop was held for all the ARPI participants. The objective of this workshop was to review and revise the issue working group reports. In addition, a set of technology working groups were established.

Throughout 1991, BBN and ISX supported several threads of activity. We collected transportation planning materials into the first "Domain Package", a set of textual documents, databases, and problem characteristics for a Joint military operation that was to be the model for IFD-2. We began to populate the CPE Repository, an on-line archive of reusable software components, demonstration systems, and domain information and data. We developed, with support from many initiative researchers, a specification and first implementation of KRSL, a knowledge representation language "interlingua" to be used to as the description language for military concepts and data to be shared among the researchers. (See The KRSL Reference Manual by N. Lehrer on the ARPI Web site.)

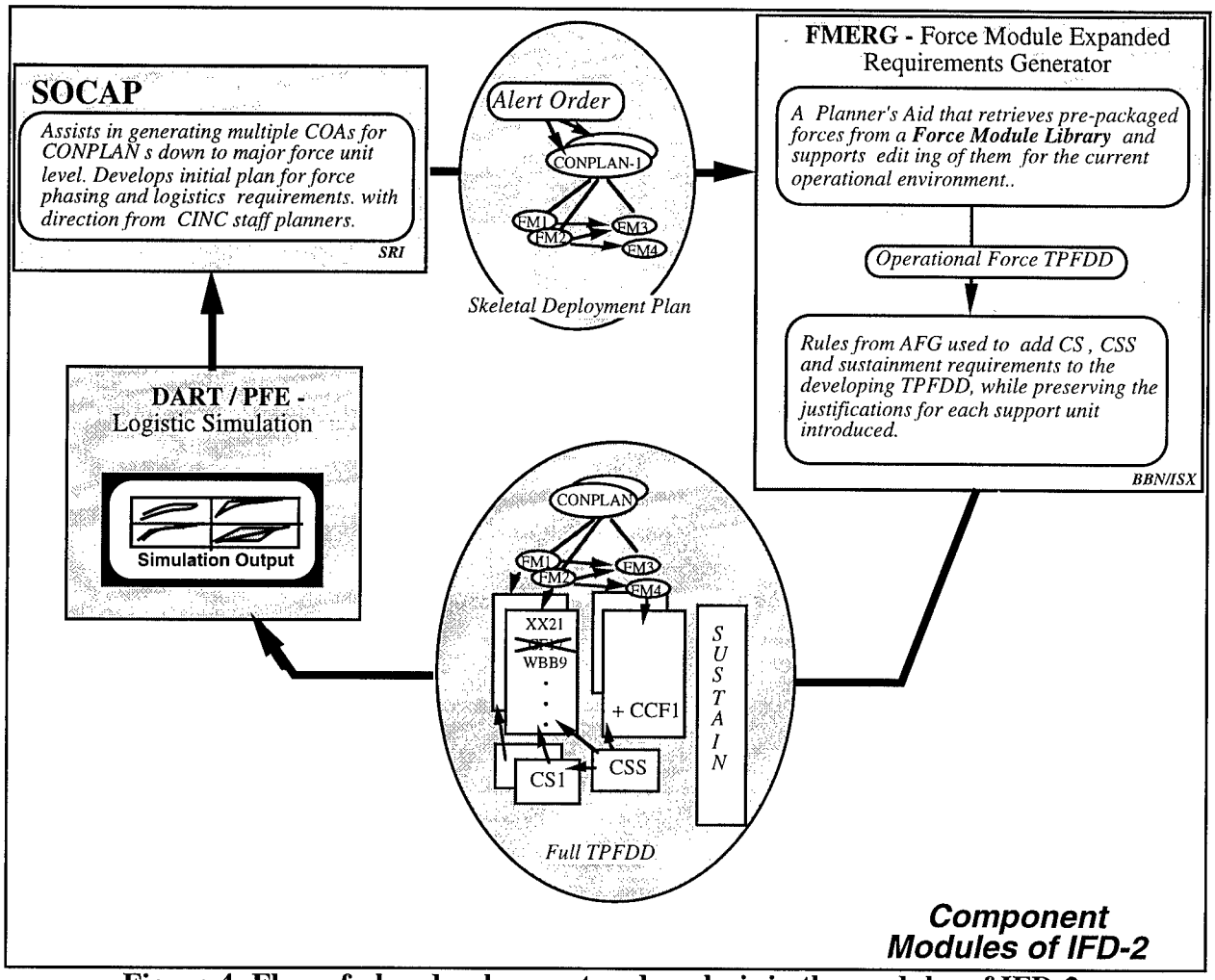

Figure 4: Flow of plan development and analysis in the modules of IFD-2.

One of the primary activities organizing much of the CPE-related effort during 1991, was working with SRI to produce IFD-2, the first IFD highlighting the potential impact of using generative planning techniques for military transportation planning. IFD-2 featured SOCAP, an application of the SIPE-2 generative planning system which was designed to help planners by automatically expanding and interleaving the activities necessary to develop a plan, while maintaining the (e.g. temporal) dependencies, that would be critical to an operation's success. SOCAP produced, with user input, a set of alternative high-level operations plans that could be expanded to a full transportation plan to test for feasibility of a proposed deployment. BBN and

ISX worked closely with SRI during the year, both collecting and providing domain information, and also developing the Force Module Expanded Requirements Generator (FMERG), a prototype force expansion tool that took SOCAP output (in the form of a list of major force units to be deployed) and produced a detailed TPFDD (for Time Phased Force Deployment Data), the military database corresponding to a transportation plan. FMERG also used a set of rules for augmenting the TPFDD with records describing the sustainment and resupply materials that would naturally flow to the theater following the initial deployment of forces. The TPFDDs produced were then analyzed for transportation feasibility using DART and PFE as part of the IFD-2 demonstration. The final demonstration took place at CENTCOM in January of 1992. Figure 4 shows the flow of control between modules for IFD-2. Appendix A contains a brief description of SOCAP and the the Force Module Expanded Requirements Generator, FMERG, as developed for IFD-2.

During the course of 1991, BBN and ISX, as administrators of the CPE Repository, also worked to collect and disseminate to initiative members the detailed textual materials, databases, and problem descriptions that were used as the basis for IFD-2. These materials, collected primarily from the Armed Forces Staff College, with much assistance from MITRE, formed the initial basis for the first "Domain Package" deposited in the CPE Repository.

Second Year Activities - Technology Integration Experiments

After the successful demonstration of IFD-2 at CENTCOM in February of 1992, BBN and ISX introduced the notion of Technology Integration Experiments as an additional means of encouraging initiative contractors to develop demonstrations that would test the potential power of integrating multiple planning and scheduling technologies together. The main emphasis in the TIEs was to promote the model where several contractors with compatible technologies would explore how to most effectively integrate their capabilities to produce a better planning system, by designing experiments which had established goals, and a 'final exam' where specific performance improvements, be they qualitative or quantitative, were anticipated and measured. The Common Prototyping Environment was ultimately developed to support such TIEs, as well as demonstration and testing of technologies; facilitate experimental system integration and evaluation activities; and capture and re-use databases, knowledge bases, software modules, and test cases.

BBN, in conjunction with ARPA and Rome Laboratory and the participating contractors, helped plan a series of Technology Integration Experiments to demonstrate interoperability between emerging initiative technologies, collect data on trade-offs between technical alternatives, and validate various technologies for potential use in an IFD. The series of TIE's, taken as a whole, also served to demonstrate a majority of the Initiative-contributed elements of infrastructure and component technology to be introduced into the Common Prototyping Environment over the next 18 months.

The TIEs were explicitly experimental: the intent was to systematically examine alternative methods of combining and utilizing a variety of technical components in radically new configurations. For example, a number of methods for exploiting temporal reasoning capabilities in the process of Course of Action Generation (COAG) were examined. Each TIE included a Final Exam in which the performance of the technical components involved were measured against some baseline criteria. The process of conducting TIE Final Exams also served to identify and prepare component technologies for insertion into IFD-3 or operational systems, as

well as benchmark performance on typical planning and scheduling problems. Table 1 lists the initial set of ties.

1. Infrastructure TIE (BBN, ISX, Unisys) demonstrated the knowledge services and distributed, knowledge-based communications elements of the envisioned distributed CPE planning environment.
2. Case Based Reasoning - Generative Planning TIE (GE-CRD, SRI) applied a CBR system developed at GE to select force units for roles in plan operators of military operations plans built by a generative planner.
3. Constraint Based Scheduling - Deployment Planning TIE (CMU, BBN, SRI) demonstrated constraint-based scheduling system in a heterogeneous military planning system. **TIE #3A** used deployment constraints to do the final stages of deployment plan development. **TIE #3B** used the scheduler for preliminary deployment plan analysis during the initial COA development, as a means of filtering the options based on transportation resource constraints.
4. Temporal Reasoning - Generative Planning TIE (GE-CRD, Honeywell, SRI) used a temporal reasoning system (Tachyon or TMM) during generative planning to propagate the temporal constraints in a developing plan. This arrangement enabled some comparative experiments with Tachyon and TMM.
5. Temporal Reasoning - Case-based Force Expansion (GE-CRD, BBN) applied temporal constraints during case-based force expansion, to ensure that units that would be available at the time they were needed.
6. Case-based reasoning - Force Expansion (GE-CRD, BBN, MITRE) used CBR techniques to refine and enumerate the elements of forces selected during high-level planning.

Table 1: The Initial TIEs

There were four general categories of TIEs:

1) **Infrastructure development**: TIEs in this category were intended to instantiate and evaluate major components of the shared CPE infrastructure needed for integrating and/or supporting planning and scheduling technical components into a cohesive system environment. (TIE #1)
2) **Component enhancement and integration experiments:** TIEs in this category were intended to investigate the value of and approaches toward enhancing technical components by composing them with others through data level interactions at various levels of granularity. Of particular importance was the determination of appropriate levels of integration and standardization and the identification and reuse of shared subcomponents. (TIEs # 2, 3, 5, 6)
3) **Tool utility experiments:** TIEs in this category evaluate the utility of general purpose tools which may contribute to one or more components. (TIEs #4, 5)
4) **System integration and evaluation experiments:** TIEs in this category were for purposes of evaluating whole system functionality, performance, and concepts of operation. In effect, IFDs represent much extended TIEs in this category. However, it was hoped that smaller scale integration experiments of this kind would spur knowledge acquisition and system design activities. (None shown.)

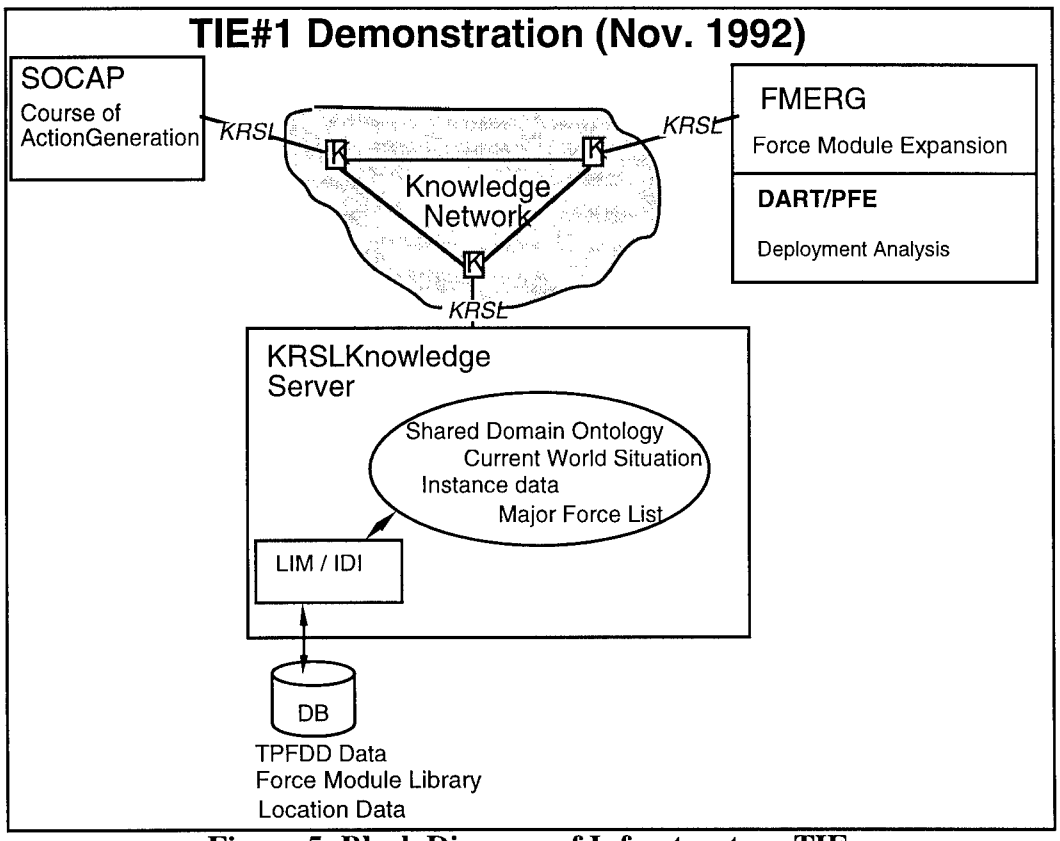

Figure 5: Block Diagram of Infrastructure TIE.

Throughout most of 1992, BBN and ISX worked with Paramax (once a unit of Unisys, now a unit of Loral) on the **Infrastructure TIE,** which demonstrated most of the architectural concepts that were ultimately used in the CPE Testbed. The goals of the Infrastructure TIE were; to provide and test infrastructure components to support a common knowledge base and intercomponent communication; to integrate KRSL, a knowledge-based distributed communications mechanism (KQML) and Paramax' Intelligent Database Interface (IDI) into a 'knowledge server', an agent in a distributed system that would answer knowledge-based queries for information from an external SQL database or from a LOOM knowledge base, and, finally; to create a knowledge level interprocess communication infrastructure for the CPE.

The Infrastructure TIE was built using most of the components of IFD-2, but replacing the file-based inter-process communications with KQML messages, and for the first time introducing KRSL syntax for all of the message content. Figure 5 depicts the Infrastructure TIE configuration. Intermediate representational products were stored in the 'knowledge server' by assertional statements in KRSL, and information needed by subsequent processes was retrieved from the knowledge server using KRSL queries. Knowledge server queries for data stored in the Oracle database were handled within the knowledge server using Paramax' LIM (LOOM Interface Module) and IDI (Intelligent Database Interface) systems, the former translating the LOOM query into an IDI format, and the latter generating the SQL query, and returning the data, which was then translated back into LOOM by LIM. The KRSL Kernel, written by Nancy Lehrer at ISX, handled the translation of KRSL queries into the corresponding LOOM query. Figure 6 shows this architecture of the knowledge server, as it was later instantiated in the CPE Testbed.

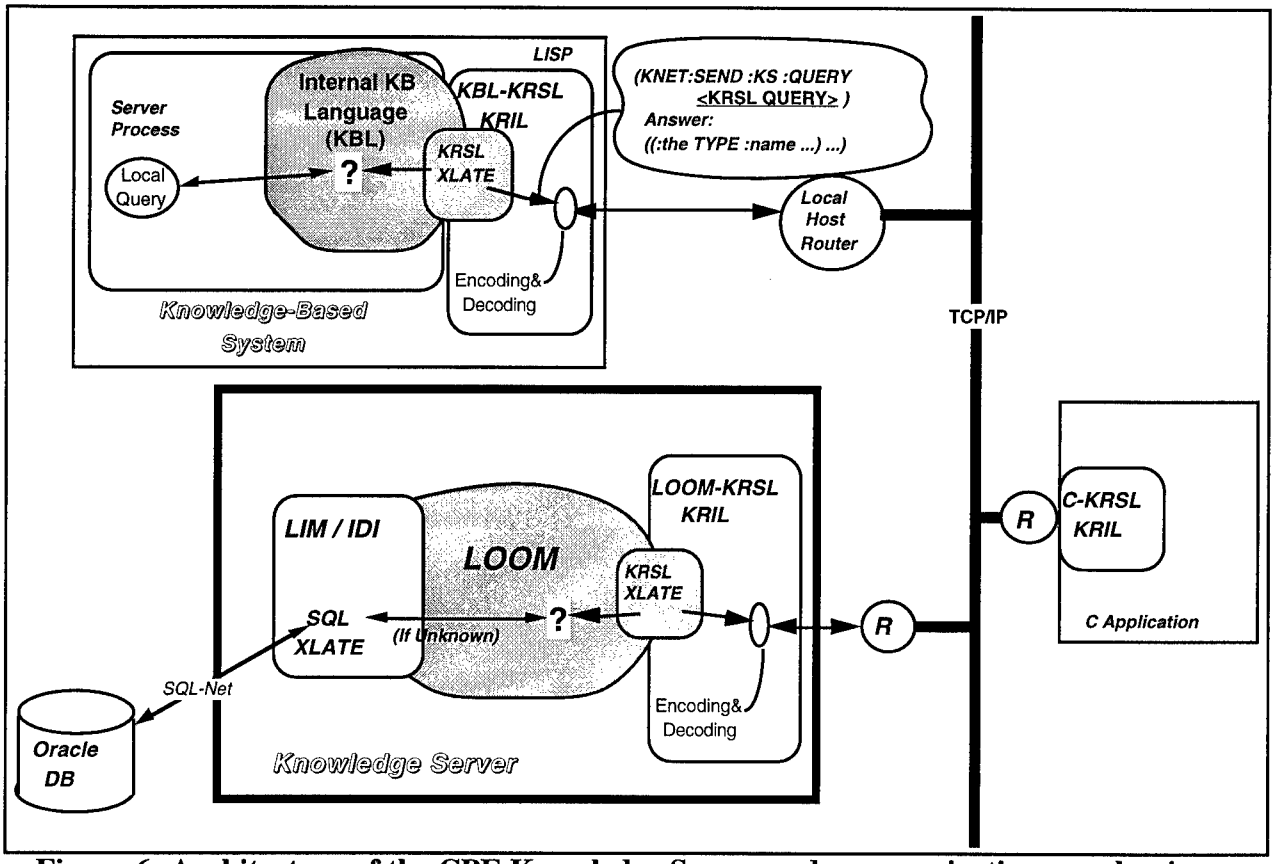

Figure 6: Architecture of the CPE Knowledge Server and communications mechanisms.

Third Year Activities - The Common Prototyping Environment Testbed

Immediately after the TIE was completed in November of 1992, work began on the CPE Testbed, which was to use many of the concepts from the Infrastructure TIE, while generalizing some of the translation and communications mechanisms. By May of 1993, when CPE Release 1.0 was demonstrated at Rome Laboratory, all of the systems that had participated in TIEs the prior year were running within the distributed CPE Testbed, using KRSL as the interlingua between modules. The various TIEs that had occurred the previous year were used as the basis for defining the functional relationships possible between the various software components that had been demonstrated. Figure 7 shows how these relationships spanned the planning functions of Course of Action Generation, Deployment Plan expansion and Scheduling and Plan Analysis. The temporal constraint managers, for example, served as plan analysis tools, providing feedback on the temporal feasibility of plans, while also propagating the temporal constraints in feasible plans to give additional guidance to the generative planner SOCAP.

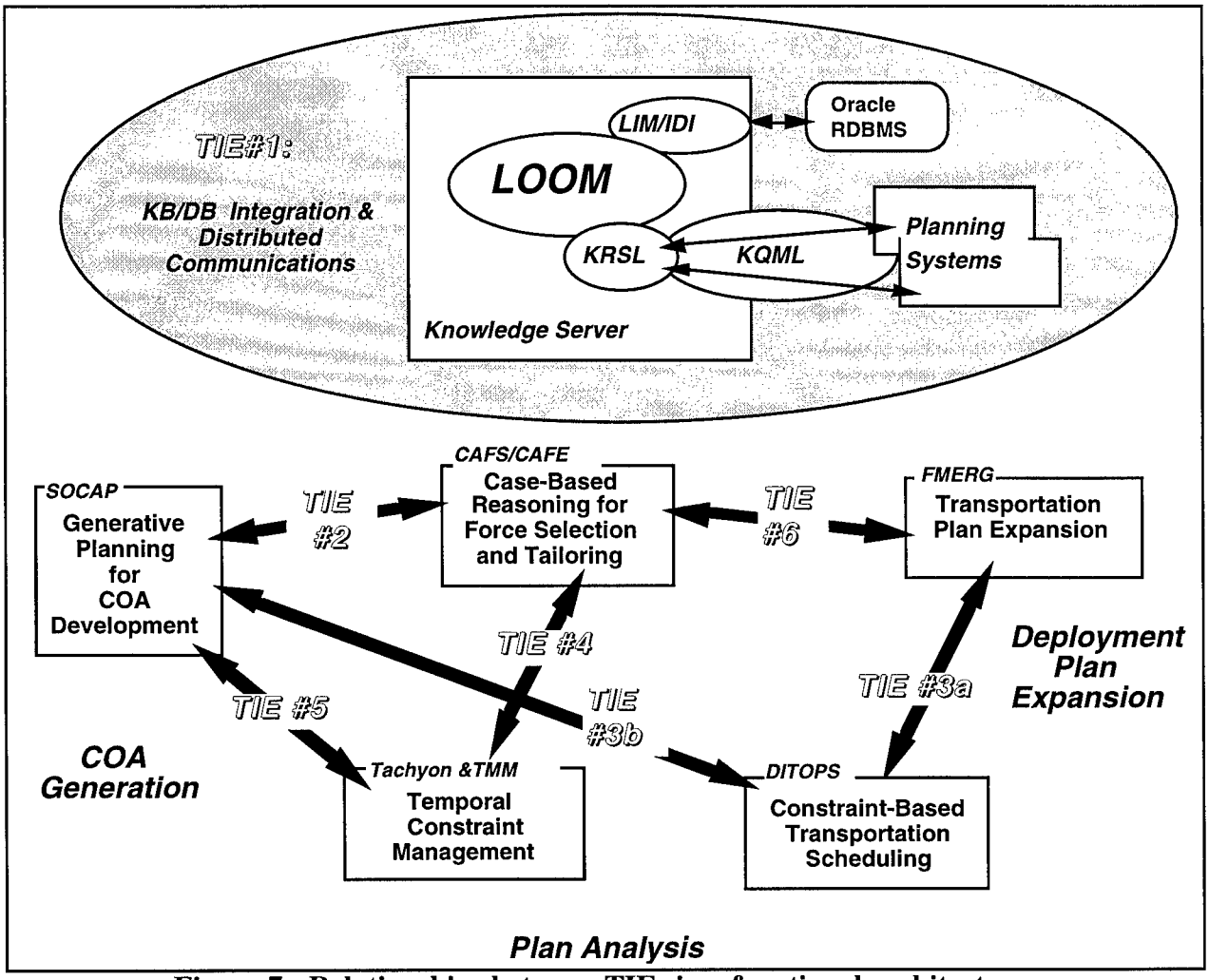

Figure 7: Relationships between TIEs in a functional architecture.

During the intervening months, several important things occurred. First, KQML, which only worked in a LISP environment at the time, was replaced by CRONUS, a more robust multi-platform, multi-language distributed interprocess communications mechanism. This permitted C applications like GE's Tachyon Temporal Reasoner to run in the same distributed environment. CRONUS also provided the functionality needed to remotely start and stop modules that could be running on any machine within the same "CPE Configuration", which might include remote sights anywhere on the Internet, such as SRI in Palo Alto, ISX in Los Angeles, Rome Laboratory in upstate New York, BBN in Cambridge, or CMU in Pittsburgh. A number of speed improvements also occurred, by using CRONUS and some data encoding techniques for list structures, we achieved an order of magnitude speedup in the time required to transmit large plan objects.

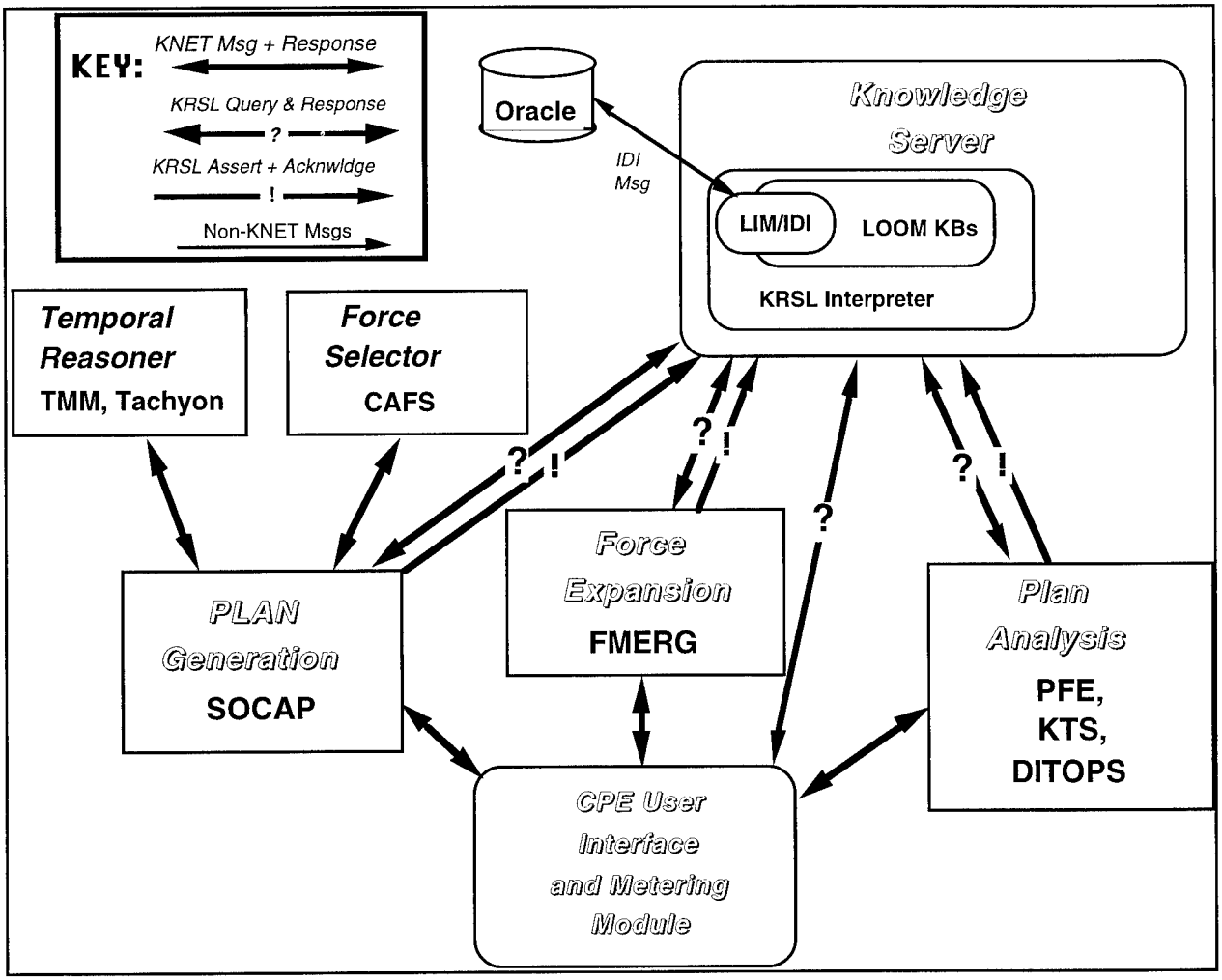

Figure 8: Communications Paths Defined between Module Types in CPE Testbed

The second major activity was defining the KRSL representations for the interprocess communications between major modules, including SOCAP, Tachyon, TMM, FMERG, PFE, and the Knowledge Server. The beginnings of a Shared Domain Ontology (SDO) were formed as representations of all of the concepts for objects reasoned over by the various modules were represented in KRSL within the Knowledge Server. (See Figure 6). These interfaces were defined based on the functionality provided by the module, so that several modules serving the same function (e.g., TMM and Tachyon, both temporal constraint managers) would use the same KRSL interface. Modules types, corresponding to the fuctionality provided by the various systems, were established, and standard message patterns were defined for each type. These communications were defined for all of the paths between module types shown in Figure 8. The messages supported by each module type are shown in Figure 9. When running experiments, one specified which software module of each relevant type was to be activated, so that comparison experiments between different technologies that could provide the same functionality were easily defined.

Current System Familiar name	SERVER ABSTRACT TYPE(s)	NICKNAME	PRIMARY OPERATIONS
Knowledge-Server	KNOWLEDGE-SERVER	KS	QUERY, ASSERT, DEFINE
TARGET	PLAN-SERVER	TARGET	QUERY
SOCAP	PLAN-GENERATOR	PLANNER	CREATE
FMERG	DEPLOYMENT-PLAN-EXPANDER	EXPANDER	EXPAND
TACHYON TMM	TEMPORAL-REASONER	TR	ANALYZE
CAFS	FORCE-SELECTOR	FS	SUGGEST-BINDING
PFE KTS	DEPLOYMENT-SIMULATOR	SIMULATOR	SIMULATE
DITOPS	" plus CAPACITY- ANALYZER		ANALYZE
ALL KNET MODULES should respond to these instrumentation operations:			SYNCHRONIZE-EXPERIMENT GET-EXPERIMENT-DATA INITIALIZE-SERVER
CLIENTS:	Description		
CPE Interface	Experiment and KNET Definition/Status Interface		
FORMAT	Force Module KA tool: Uses FM Library accessed thru Knowledge Server		

Figure 9: Operations defined for Module Types (as of Release 2.0)

A major portion of this effort was in defining the KRSL equivalents for plans, as produced by SIPE/SOCAP in a more generic form so that other planners might be expected to generate similar plans. Figure 10 shows the representation of a typical node in a SOCAP-produced plan, as represented in KRSL for transmission to the Knowledge Server. Figure 11 shows the KRSL content of messages between the generative planner SOCAP and the Case-based force selection module CAFS.

```
(:the plan :name p6120
    :documentation "a node in a plan"
    :method DETER-THREAT-BY-SHOW-OF-FORCE  ;; operator
    :parameters ((opposing-force :VALUE (:the FORCE :name 1ST-ARMBDE))
                 (situation :VALUE (:the ENEMY-COA :name ENEMY-COA-1)))
    :goal (DETER-IMMEDIATE-THREAT 1ST-ARMBDE ENEMY-COA-1 21)
    :in-service-of (PROTECTED-TI  ENEMY-COA-1)
    :effects (DETER-IMMEDIATE-THREAT 1ST-ARMBDE ENEMY-COA-1 21)
    :let (CDAY 23)  ;; latest end time
    :expansions  (:the PLAN-EXPANSION :name EXP101
                       NODES (:values (:the plan :name P6390)
                                      (:the plan :name P6391)
                                      (:the plan :name P63192))
                       GRAPH (:values
                                      (INTERVAL-BEFORE P6390 P6391)
                                      (INTERVAL-BEFORE P6391 P6392)))))
```

Figure 10: A KRSL Plan Node

PLANNER Sends:

```
(knet:SEND :force-selector  :suggest-binding
   :for-arg 'FRIENDLY-UNIT
   :in-plan
        (:THE PLAN :NAME P6158
            :METHOD DETER-BORDER-INCURSION-BY-GROUND-PATROL
            :PARAMETERS
                ((FRIENDLY-UNIT :TYPE-RESTRICTION ARMY)
                 (ENEMY-UNIT :VALUE (:THE ARMORED :NAME ALG-1STARMBDE))
                 (COA :VALUE (:THE ENEMY-COA :NAME ENEMY-COA-1))
                 (ROUTE :value (:the ROUTE :name ROUTE-1))))
            :CONSTRAINTS
                (:and (can-traverse (friendly-unit p6158) (route p6158))
                      (greaterp (firepower (friendly-unit p6158))
                                (firepower (enemy-unit p6158))))
        )
```

Force Selector returns a list of possibilities like:

```
(PARAMETER-BINDING P6390 'FRIENDLY-ARMY (:the IMB :name IMB-1889)
                '*COMPLETE-MATCH*)
```

Figure 11: Communications between COAG Planner and Force Selector

This facilitated comparative experiments, such as a direct comparison of TMM and Tachyon supporting SOCAP as it built the same plan structures. Figure 12 shows the modules of the CPE Testbed involved in such an experiment. From the CPE Experiment Control User Interface, two trials are set up, one in which SOCAP is called to build a plan using TMM for temporal reasoning support, the other using Tachyon. Each trial is metered such that the timings of the various processes are collected (SOCAP calls the temporal reasoner twice during each level of plan expansion). The results are collected from each module at the end of each trial, and sorted to form tables of data that can be analyzed using the CLASP statistical analysis system (provided by UMASS).

Figure 13 shows the CPE Experiment Control interface from which such experiments were defined. The displayed menus cover defining experiments and trials, collecting data for analysis by CLASP, means of browsing the knowledge server, defining and managing the software modules within the distributed CPE network configuration, and other miscellaneous commands. See the CPE Users Manual, Reference Guide and Installation Guide for complete descriptions of all of the commands shown.

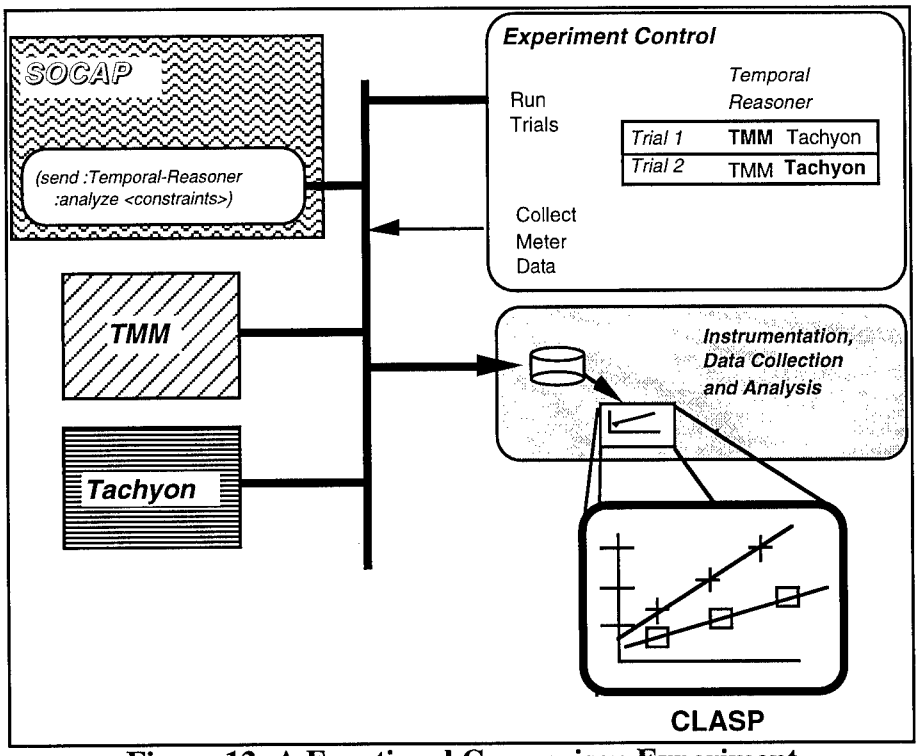

Figure 12: A Functional Comparison Experiment

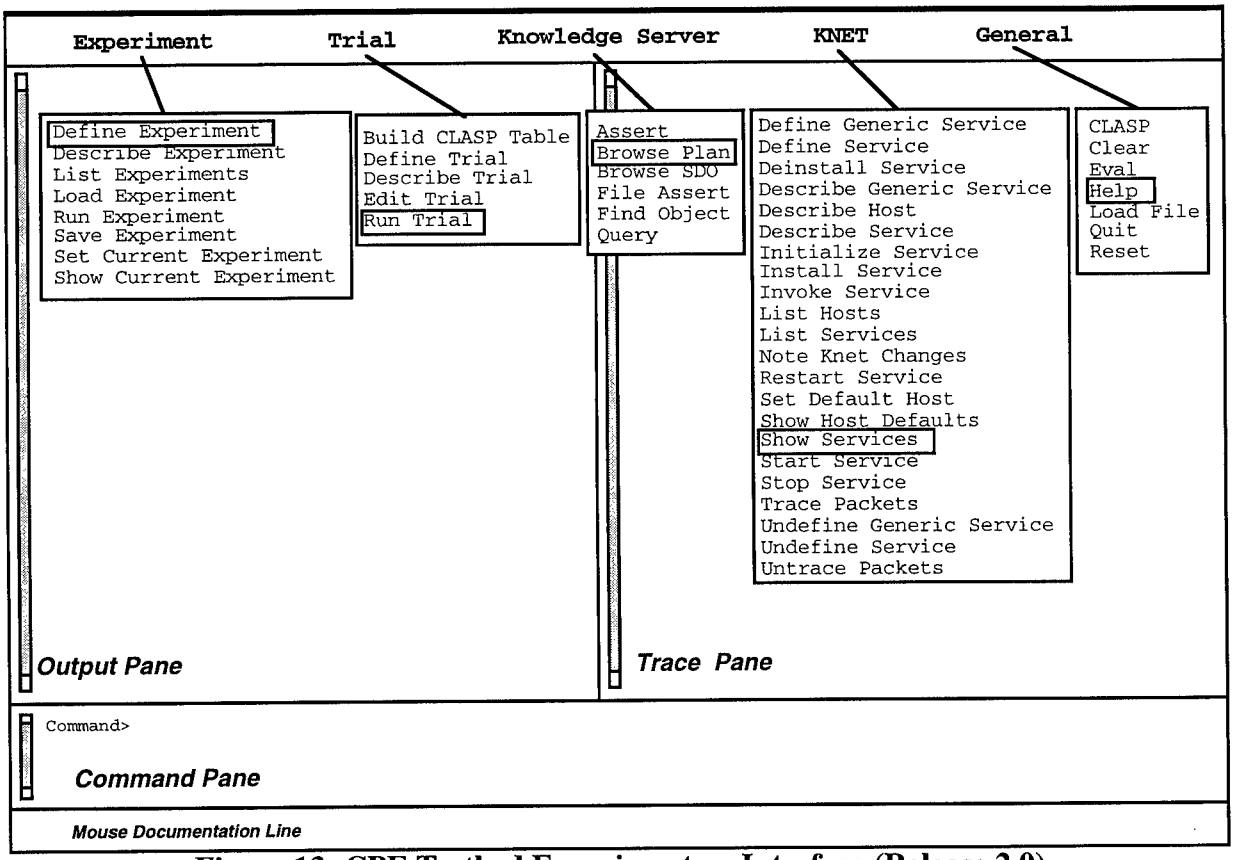

Figure 13: CPE Testbed Experimenters Interface (Release 2.0)

CPE Testbed Release 1.0 was demonstrated at the end of May, 1993 at Rome Laboratory. It involved modules running at Rome Laboratory, BBN Cambridge, Paramax in Philadelphia and ISX in Atlanta. The CPE was first presented to the ARPI community at a quarterly meeting that took place during the AAAI conference in Washington DC during July of 1993.

CPE Testbed Release 2.0

After the first release of the CPE Testbed, work began on a second release that addressed some of the outstanding issues relating primarily to the use of the CPE Testbed by new research groups. The issues included:

- Ease of installation of the CPE Testbed at a new sight, and its configuration for use with other CPE Testbed sites.
- The reduction in size of the installed systems, and the development of a 'CPE Lite' installation.
- Generalized mechanisms for "wrapping" modules with pre-defined KRSL translators. The development of translators for LISP/CLOS, and for modules written in C++.
- Mechanisms for asynchronous communications and multi-casting.
- Improved mechanisms for defining and controlling experiment meters (data collectors) at arbitrary points within the source code for a module, and better means of controlling those meters during experiments.
- Improvements to the meter data collection system, and the automatic generation of CLASP compatible data files so that the data could be analyzed using simple commands from the CPE and CLASP interfaces.
- The generation of a complete set of installation, reference and user manual documents.

This work continued throughout the remainder of 1993, and CPE Testbed 2.0 was formally released at the ARPI workshop that occurred in February of 1994.

As an example of the new capabilities of the CPE Testbed, a demonstration was given that included a 'Plan Server', represented as a Knowledge Server module type. This plan server was built by defining a module that could access the TARGET object database and answer requests for plans that had been built using the TARGET Plan Editor interface, using a query identical to one that would retrieve a plan from the original Knowledge Server. The TARGET Plan Server module was written in C++ to access the C++ based Objectivity ™ OODB. For the demonstration, the FMERG Force Expansion module was called on to expand a plan. This caused an asynchronous, multi-cast request to go out to all Knowledge Servers that might contain the high-level major forces plan named. Since both the original Knowledge Server and the new Plan Server were active during the experiment, both would be available to answer the query, but only the Plan Server had the plan named. It's response, in KRSL, looked syntactically identical to one that might have come from the Knowledge Server. Figure 14 shows this demonstration.

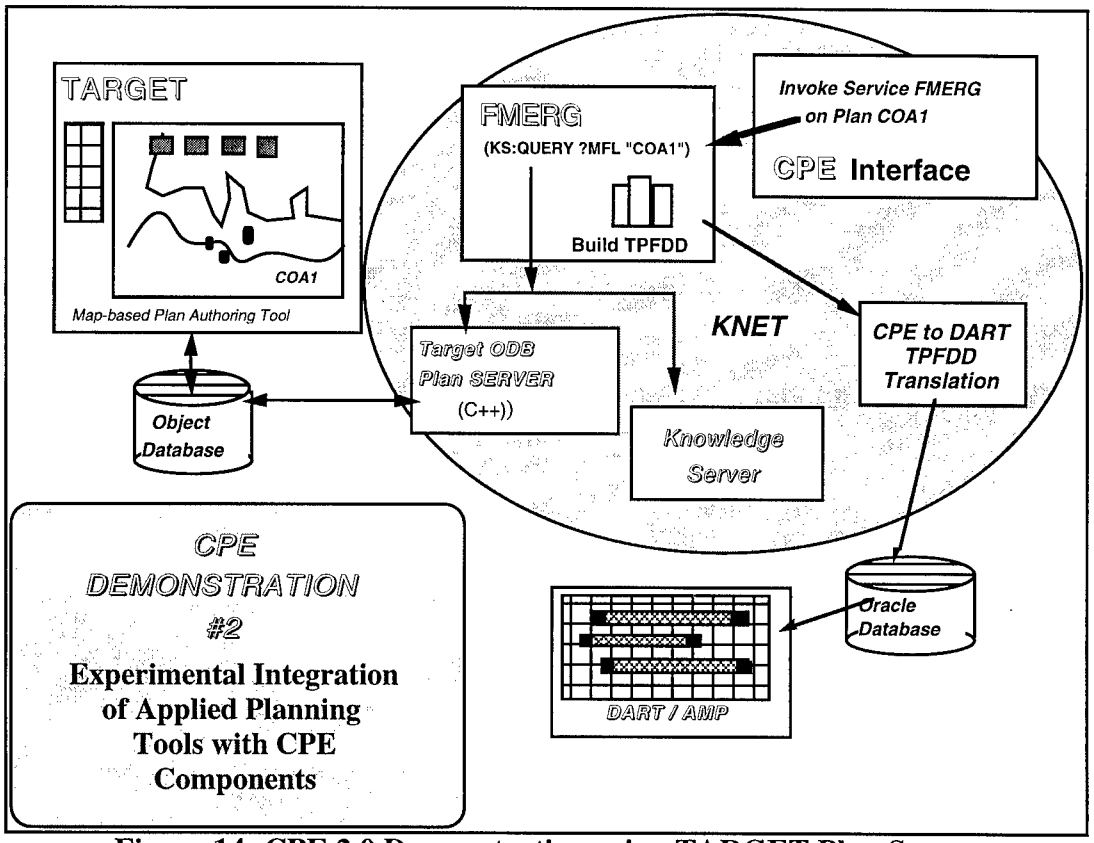

Figure 14: CPE 2.0 Demonstration using TARGET Plan Server

In another demonstration of the new capabilities of the CPE, an experiment was defined in which a whole set of trials were defined at once, by building the cross-product of sets of alternative modules, and sets of alternative inputs. The CPE Testbed Experimenter's interface took care of the process of correctly initializing each trial, and organizing the resulting metering data into tables that could be used for a scatterplot or regression analyses. Figure 15 shows the structure of this experiment and gives a sense of the interfaces involved.

The 1994 ARPI Workshop

One of the responsibilities of the CPE team was the organization of the annual ARPI Workshops, including participation in most of the working groups that met at quarterly meetings and especially at the annual meetings in which all ARPI contractors participated. The 1994 ARPI Workshop, which took place in Tuscon, Arizona in February of 1994, was organized somewhat differently in that it had three parallel technical paper sessions in addition to several working groups.

Mark Burstein of BBN served as the program chair for the paper tracks and edited the 500+ page proceedings of that workshop, distributed by Morgan Kaufman Publishers. He also served as co-chair, with Prof. Drew McDermott, of the working group on Mixed-Initiative Planning, which had met first at Yale University at the preceding ARPI quarterly in November of 1993. The report of the Mixed-Initiative Planning working group was published in the 1994 ARPI Workshop Proceedings (Burstein, 1994), and an edited version (Burstein & McDermott, 1996) appeared as a chapter in the book*Cognitive Technology* (Gorayska and Mey, 1996).

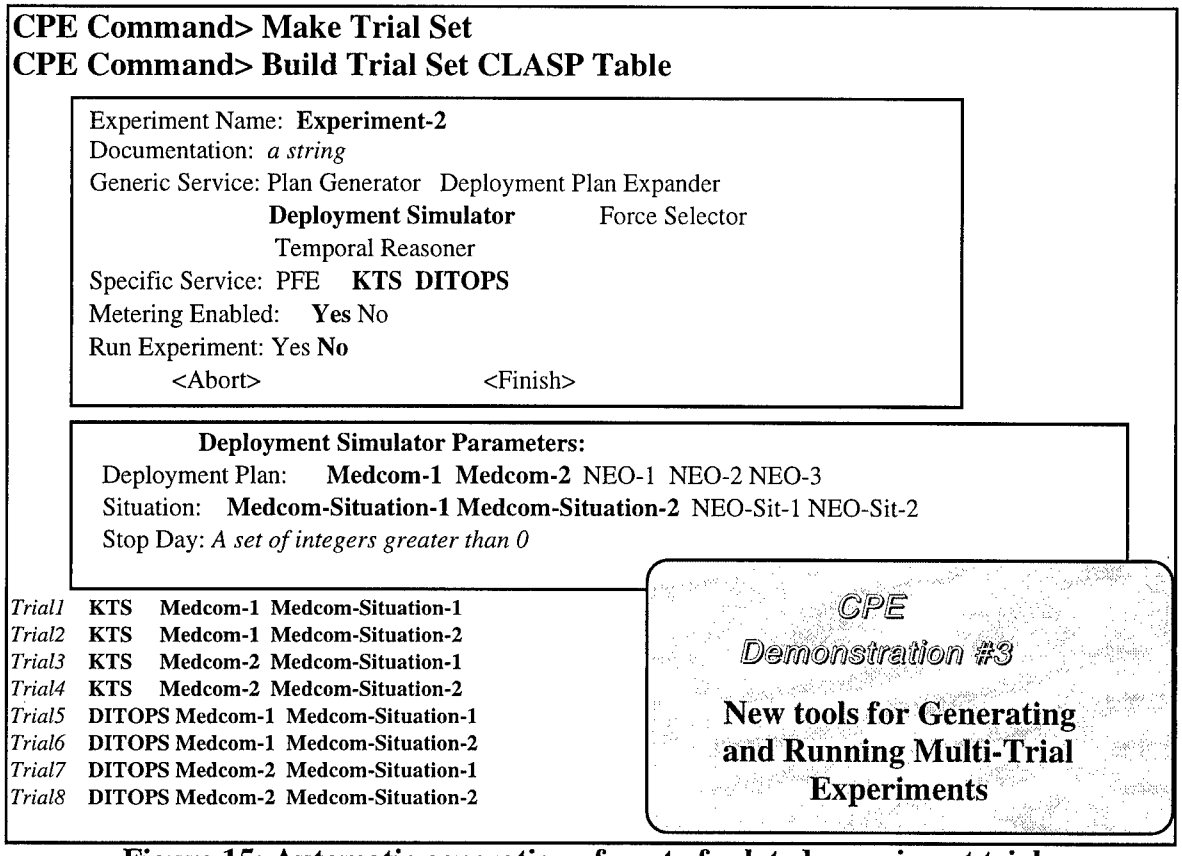

Figure 15: Automatic generation of a set of related experiment trials.

Fourth Year Activities - New CPE Modules

During 1994, a number of additional modules were added to the CPE Testbed. In addition to the TARGET Plan Server, described above, the ForMAT Case-based Force Module Acquisition Tool developed by MITRE was incorporated into the CPE, as a way of building and re-using force modules for deployment plan expansion. DITOPS, the constraint-based scheduling tool from CMU was given a second function as a high-level resource analysis tool in support of COAG planning (e.g. using SOCAP).

In the area of data access, CoBase (UCLA) and SIMS (ISI) were connected in a TIE that enabled the SIMS query planning system to find results even when not precisely the desired data was available, using CoBase's inexact query mechanisms. These systems were demonstrated using some CPE component technologies.

Fourth Year Activities - Publications and Domain Models

Also during that year, an ARPI web site was introduced, as an alternative means of distributing information to initiative members and the general public. Documents of general interest, such as KRSL language specifications, were made available by this means.

A paper and videotape of the CPE Testbed were produced. The videotape describes the mechanisms of the CPE and runs through several experiments as demonstrations. The paper, which appeared in IEEE Expert issue for February 1995, is included as Appendix B.

During the third and fourth years, ISX worked with the University of Edinburgh, MITRE and USC/ISI on a 'toy domain' model for research on military planning. This model was built around a fictional island called 'Pacifica', shown in Figure 16. A paper describing this environment, dubbed PRECiS, is available as (Reece et al, 1993) from the ARPI Web site. ISX also encoded this domain model in KRSL and placed it in the CPE Repository for access by ARPI members.

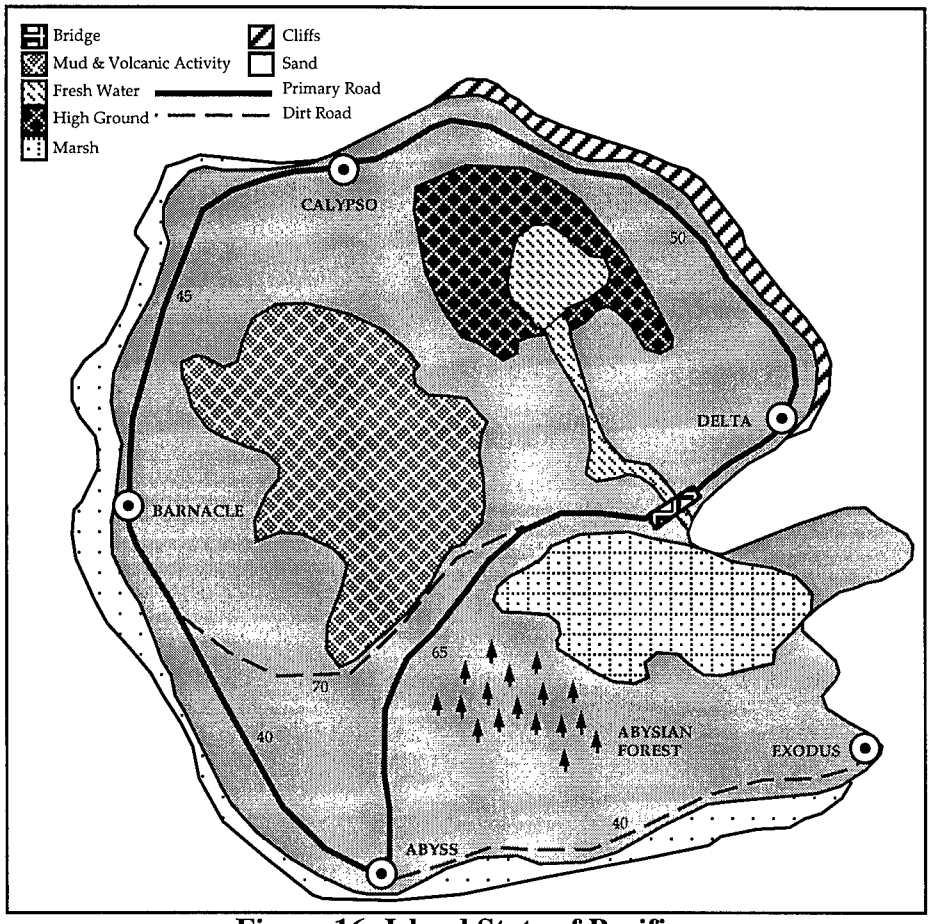

Figure 16: Island State of Pacifica

Fourth Year Activities - ITAS

During the final year of the contract, BBN and ISX each engaged heavily in technology transition activities. BBN, working with Kestrel Institute, developed the In-Theater Airlift Scheduler (ITAS), as an application of Kestrel's semi-automatic program generation system KIDS which had previously built KTS, a strategic air and sealift scheduler similar to DITOPS and based on the model provided by PFE. ITAS, in contrast, was built with the close cooperation and support of the airlift schedulers commanded by LtC. Stuhldreher at PACAF, the US Airforce Pacific Command. It was built as an operational prototype that could be used in the field by PACAF, and so was designed to run on the Apple Macintosh ™ hardware that was used in that command. ITAS is much closer to a real-world scheduling application than previous schedulers generated under ARPI funding had been. It enables airlift schedulers to do in minutes what used to take hours each day, and permits them to look at a much wider variety of

alternatives than they were previously able to do. ITAS was successfully demonstrated in exercises at JWID 1994 and JWID 1995. It is described in detail in (Burstein and Smith, 1996), which appears as Appendix C.

APPENDICES

PUBLISHED REPORTS AND OTHER ARPI AND CPE RELATED DOCUMENTS

The appendices to this report contain additional important material.

Appendix A contains a summary of the Force Module Expanded Requirements Generator, FMERG, that was used in IFD-2 and in the CPE Testbed.

Appendix B contains a paper describing the Common Prototyping Environment, as published in (A. Tate, Ed.) *Advanced Planning Technology* , The AAAI Press, 1996.

Appendix C contains a paper describing the In-theater Airlift Scheduler, ITAS, to appear in (B. Drabble, Ed.) the Proceedings of the Third AI Planning Systems Conference, Edinburgh, Scotland, The AAAI Press, 1996.

Other papers relating to the CPE project can be obtained via the ARPI Web site (http://arpi.isx.com) or by contacting Dr. Mark Burstein at BBN (burstein@bbn.com) or Mr. Mark Hoffman at ISX (mhoffman@isx.com):

Burstein, M. (February 1991) Software Specification of the CPE, with an appendix describing the domain of the Military Transportation Planning. (BBN TR No. 7495)

Burstein, M. and Kosy, D. (April, 1991) The Prototype Feasibility Estimator (PFE). BBN Technical Report No. 7598, a description of a LISP-based simulator developed by BBN and CMU to model strategic transportation, based on the RAPIDSIM model used in DART.

Reece, G., Tate, A. Brown, D. and Hoffman, M. (August, 1993) The PRECiS Environment.

Burstein, M. (Ed.) (1994) .) *The 1994 ARPI Workshop Proceedings.,* Morgan Kaufman.

Burstein, M. and McDermott, D. V. (1994) "Issues in the Development of Human-Computer Mixed-Initiative Planning Systems" In M. Burstein (ed.) *The 1994 ARPI Workshop Proceedings.,* Morgan Kaufman. " (A similar paper appears in (B. Gorayska and J. Mey, Eds. *Cognitive Technology*., Elsevier, 1996.)

Lehrer, N., et al. The KRSL Reference Manual. (latest revision)

Hoffman, M., et al. The Shared Domain Ontology. (latest revision)

APPENDIX A:

Developed for **IFD-2** (1992)

and adapted for use in the CPE

A Brief Summary of the role of the Force Module Expanded Requirements Generator (FMERG) in IFD-2 (February, 1992)

Mark Burstein (BBN) and

Marie Bienkowski (SRI International)

January, 1992

1. INTRODUCTION

This is a brief description of the FMERG component of the January 1992 integrated feasibility demonstration (IFD-2) that was produced in support of the DARPA/RL Knowledge-based Planning and Scheduling Initiative. Part of this initiative, which includes work that will advance the state of the art in artificial intelligence (AI) techniques for planning and scheduling, is to demonstrate yearly the feasibility of applying existing or newly-developed technologies to the problem of planning and executing joint military operations. Particular emphasis has been placed on transportation planning as the target problem; thus it is envisioned that the US Transportation Command (TRANSCOM) could make use of or benefit from the technology or applications developed in conjunction with planners at other CINCs.

The main purpose of the integrated feasibility demonstrations is to show the end user community -- military planners -- the ability of state-of-the-art technology (including AI and conventional software) to significantly impact their operations. The demonstration will show the possibility of improvements in their existing methods, either by shortening the time it takes them to plan or by improving the quality and quantity of the plans produced. The IFDs might also suggest to the end users ways in which they might alter the way they plan to exploit these improvements.

As part of the planning initiative, SRI is applying a knowledge-based generative planning system (called SIPE-2) to the problem of generating joint operations plans. The system, SIPE-2 for Operations Crisis Action Planning (SOCAP), will act as a decision aid to enable both employment and deployment actions to be generated given a mission, constraints, and guidance. SOCAP is targeted for the J3 staff (operations) at a joint command such as US Central Command (CENTCOM).

The main focus of SOCAP is the development of the operations plan and estimations of its feasibility on a number of features. The planning technology underlying SOCAP could also be applied to generating other parts of the complete plan needed to execute a mission, such as the logistics or intelligence plan. SOCAP was designed to aid an operations planner in determining that a generated course of action (COA) is transportationally feasible. As users develop a COA using SOCAP, they can make use of a transportation feasibility estimator, such as the ones in DART, to check their plan. The integrated system is designed so that later versions will permit the insertion of different types of feasibility estimators, e.g., for logistics, command and control, etc. Furthermore, since the underlying generative planning technology (SIPE-2) develops a plan at successively finer levels of detail, it may be possible to estimate the feasibility of the plan at different levels as well.

Figure 1 shows thefunctional impact of the use of SOCAP and DART, as well as the connection between them. SOCAP generates a plan which includes both employment actions that accomplish a mission and deployment actions that move the forces referenced in the employment actions to the theatre where they will be employed. DART uses the TPFDD data derived from the deployment actions to assess the transportation feasibility of the operations plan. Shortfalls are fed back to the SOCAP users, who then modify the plan to overcome the transportation problem. The gap between the plan generated by SOCAP and the TPFDD input expected by DART will be bridged by the FMERG module, which performs functions similar to those of the Auto Force Generator (AFG) and Movement Requirements Generator (MRG). Essentially, FMERG must take the skeletal plan produced by SOCAP and create a standard TPFDD that can be tested by DART using one of the transportation feasibility estimation simulators, RAPIDSIM or PFE. For IFD-2, PFE was extended to return information on transportation feasibility to SOCAP, to enable some replanning to occur.

2. FMERG

The Force Module Expanded Requirements Generator module in the 1992 Integrated Feasibility Demonstration (IFD-2) provides a "vertical slice" thru the process of developing a TPFDD from a list of major forces to be deployed. As such it is a placeholder for several more sophistocated components in our vision of a highly automated transportation planning and scheduling system of the future. FMERG expands force modules and inserts their component units into a transportation plan, by drawing on a library of hierarchical Force Module descriptions. It takes as its primary input a high-level plan produced by SOCAP, and extracts from that plan the deployment requirements for the major force units referenced in the plan. Using its library of hierarchical descriptions of force module compositions, it developes a TPFDD with requirements to deploy all force module elements from their home bases to the operations staging areas specified in the SOCAP plan. The TPFDD produced includs relative temporal constraints on the deployment of those units. It augments this basic TPFDD plan with additional deployment requirements for Combat Support, Combat Service Support and Sustainment elements. The main product of the FMERG module is a TPFDD embodying the decisions made by the high-level planner (The SOCAP automatted planning system was used for this purpose in IFD-2) about the deployment of major force units.

The primary components of IFD-2 are shown in figure 2 below. Overall, the demonstration highlights the semi-automated plan generation capability to be embodied in SOCAP, and support its use in developing a feasible TPFDD by connecting it to DART thru simple versions of two other tools which together formed the FMERG system: a Force Module Refinement tool (FMR), and a Support Force Generation (SFG) tool. The latter used rulesets like those developed for two JOPES systems: AFG, the Auto-Force Generator, and MRG, the Movement Requirements Generator. Within DART, the PFE deployment simulation tool (a model much like RAPIDSIM) was used to estimate the feasibility of the TPFDD generated.

Ultimately, we would expect information about closure to be be fed directly back to SOCAP to support plan revision.

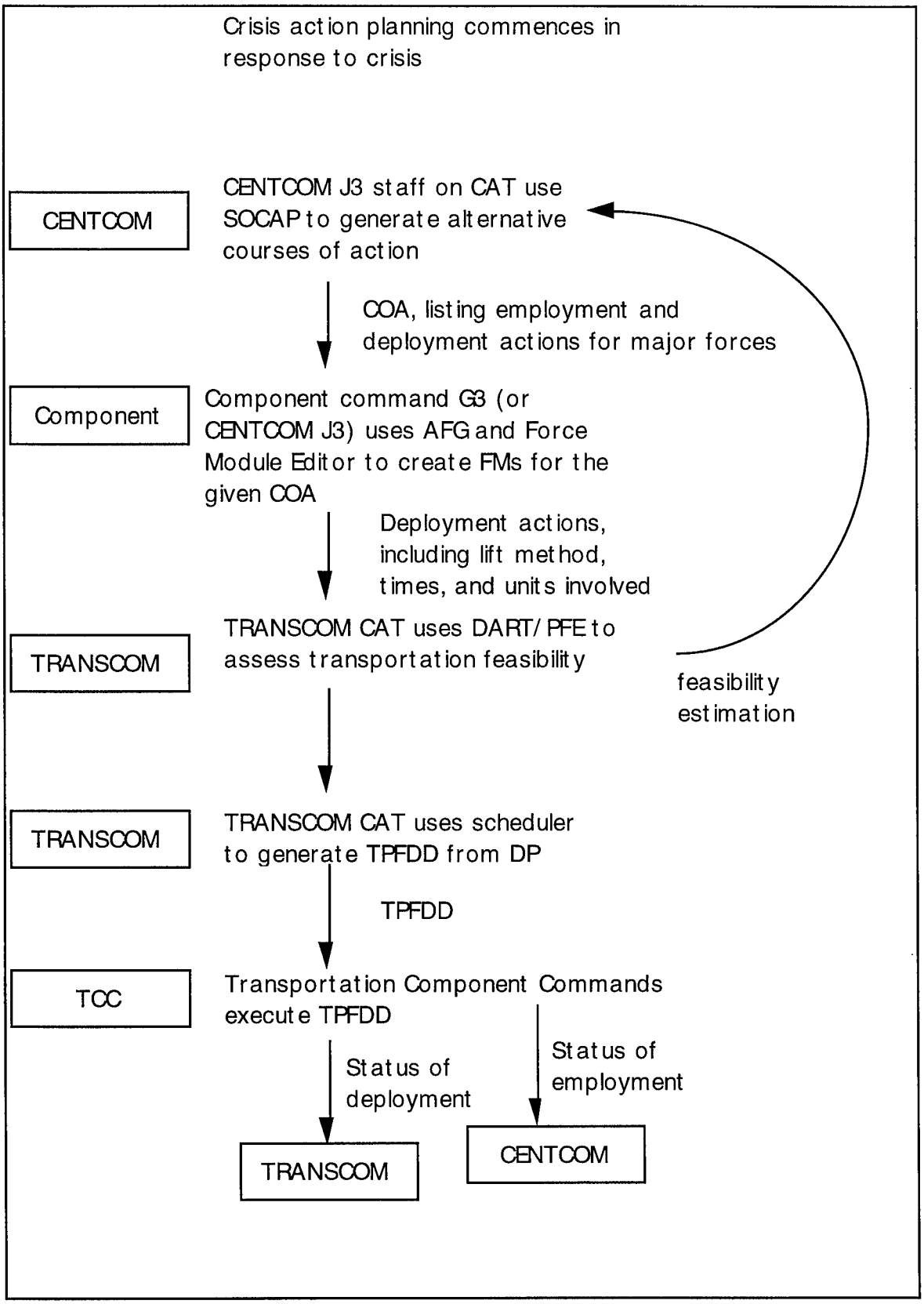

Figure 1: IFD-2 Functional Flow

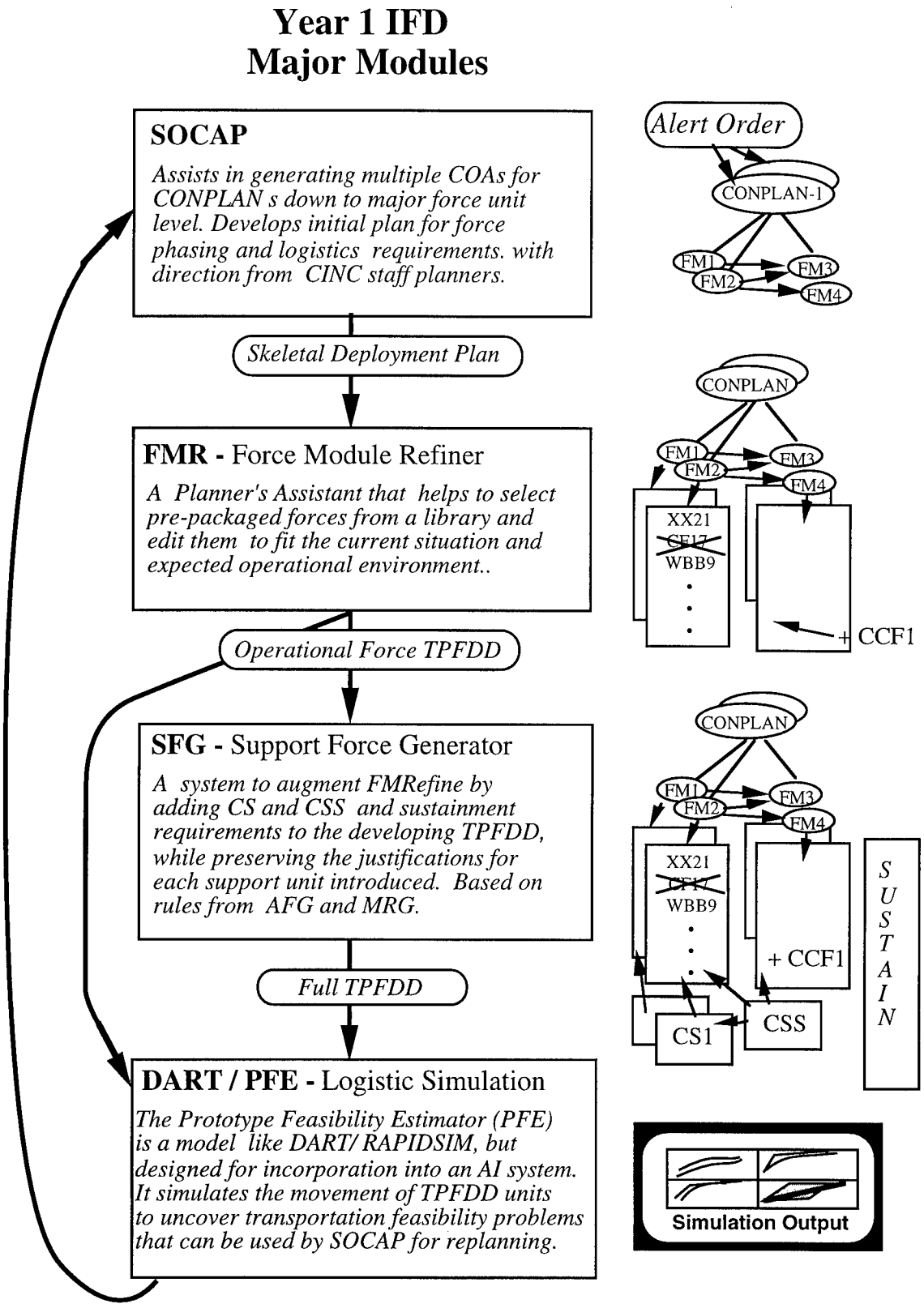

Figure 2: Major Modules in IFD-2.

The Force Module Library that was used in the demonstration was based on that utilized at the Armed Forces Staff College (AFSC) in their training course exercise on Joint Military Planning, with additional information included about the inter-unit constraints on unit deployment to support an employment concept. The FM library used contained approximately 35 FMs from all branches of the service and from various force levels from Corps down to Brigade.

A specially adapted TPFDD editor was included with FMERG to allow the user to modify and further adapt the Force Modules to be used in the plan. A procedural interface will allow both the planning system and the human planner to manipulate {augment or reduce} the detailed force list prior to passing it back to SOCAP or other modules.

At this point in the normal planning process, additions for supply, resupply, and buildup (CINS and PINS) are made. These additions were developed in FMERG based on a set of adjustable planning, supply, casualty, and resupply factors similar in nature to those supported by the AFSC Joint Operation and Planning System (JOPS) in the Auto Force Generator (AFG) and Movement Requirement Generator MRG. Sustainment CINS and PINS were added using a variety of standard planning factors modeled after those in LOGGEN.

APPENDIX B:

The Common Prototyping Environment:
A Framework for Technology Integration, Evaluation and Transition

- as published in *Advanced Planning Technology* (A. Tate, Ed.)
The AAAI Press, Menlo Park, CA, May 1996.

The ARPA/Rome Laboratory Planning Initiative

Common Prototyping Environment:

A Framework for Technology Integration, Evaluation and Transition

Mark H. Burstein, Richard Schantz
BBN Systems and Technologies

Marie A. Bienkowski, Marie E. desJardins
SRI International

Steven Smith
CMU Robotics Institute

INTRODUCTION

The ARPA and Rome Laboratory sponsored Planning Initiative (**ARPI**), is a government funded research program that began in 1989. Its long range objective is to develop large-scale distributed, collaborative, automated and semi-automated military planning and scheduling systems. At the outset of the initiative, BBN Systems and Technologies and ISX Corporation were teamed to provide in the form of a Common Prototyping Environment (CPE), and aid the Initiative's research community by providing access to realistic military planning problems, while reducing each individual project's overhead by providing a suite of sharable software tools that would foster the integration, evaluation and transitioning of the planning and scheduling technologies developed. The CPE Testbed, the primary focus of this article, was developed in 1993-4, as a distributed environment for Technology Integration Experiments that combined the planning and scheduling systems developed by the community to address specific kinds of problems.

The goals of the Common Prototyping Environment project are:

- To foster the development of reusable planning and scheduling technologies by providing access to useful supporting software and development tools and sharable domain knowledge and data,
- To support the performance of experiments in which these technologies are combined to solve planning problems of interest to the military, and
- To ease the transitioning of technologies into military operational prototype systems through demonstrations of interoperability.

The CPE project is a team of people dedicated to these tasks. BBN Systems and Technologies and ISX Corporation were teamed together to develop and support the CPE. BBN has had primary responsibility for the development the CPE system and some of the supporting software systems used in integration experiments, and ISX has been primarily responsible for providing access to domain data and knowledge to support the Initiative's researchers and specific integration efforts and experiments.

This article describes the major activities associated with the development of the CPE. We deal separately with the two major aspects of the CPE project as a whole. **The CPE Repository** is an "electronic clearinghouse" for software (sources, executables, and sometimes both) developed under ARPI auspices, and representative data and knowledge, in both textual and electronic

forms, about the military transportation planning processes. **The CPE Testbed,** developed in the last two years of the project, is a suite of software tools to support the integration of and experimentation with prototype planning and scheduling systems in a distributed communications environment.

There have been a number of collaborative project activities throughout the course of the Planning Initiative that have helped to foster cooperative development of software demonstrations and the experimental integration of technology components. **Integrated Feasibility Demonstrations (IFDs)** (see accompanying article, this issue) were joint projects to integrate existing state-of-the-art technology components and demonstrate their potential to greatly improve on the time or quality of planning within the military. In addition, smaller-scale, **Technology Integration Experiments (TIEs)** were developed so that different research and development teams could explore the issues involved in successfully combining their research software.

IFD's have had a much stronger operational focus than the TIEs. IFD-1, which produced the DART system, was developed just before and during Operation Desert Shield, and was in active use almost immediately. IFD-2 and IFD-3 were developed primarily as demostrations, built around a different, specific type of military planning situation. IFD-2 was focused on strategic and transportation planning for a "typical" small scale, primarily defensive, military operation, while IFD-3 was developed using as a test-scenario a non-combatant evacuation operation or NEO. The unclassified data and textual materials describing the scenarios used in each of these demonstrations was captured and made available to the ARPI research community as part of the CPE Repository, as described in Section 2.

The series of Technology Integration Experiments or TIEs began shortly after IFD-2 concluded in February of 1992. The initial set of TIEs focused on integrating more researchers technologies into a distributed software environment that included the IFD-2 demonstration prototypes. These TIEs addressed a range of related transportation planning and scheduling problems grounded in the IFD-2 scenario. One additional TIE, called the "Infrastructure TIE" focused on applying the techniques for database and knowledge-base access and for distributed, knowledge-based systems inter-communications (KQML) being developed under ARPA sponsorship. For that TIE, BBN and ISX worked with Paramax Corporation (now Unisys) to develop the initial version of a software infrastructure that became the core of the CPE Testbed, including the first prototype of a "knowledge server".

Release 1.0 of the CPE Testbed in May, 1993 was, first and foremost, an engineering effort capitalizing on all of the TIEs conducted during 1992. It brought all of the planning systems that had used in the TIES "together", so that they all worked in the same distributed environment, communicating among themselves using a consistent set of distributed software communications protocols (KNET, a simple variant of KQML) and using a single knowledge representation language for information exchange (KRSL).

The CPE Testbed was also designed to support experiments in which components were combined in novel ways, or directly compared doing similar functions. To address the issue of experimentation, the testbed includes a suite of tools for collecting and analyzing data about the performance of the modules used in solving planning problems. Comparative experiments are readily conducted in the CPE by substituting one software system providing a defined type of service for another, and collecting the same performance measures for both systems. The data that is collected can be analyzed using the CLASP statistics package, provided by the University of Massachusetts, all from within the CPE Interface.

The CPE Testbed now includes technology components developed in more than ten different R&D laboratories sponsored under the initiative. These software systems all provide services that can be utilized in experiments based on solving military planning problems, even when the software systems are physically hosted at sites dispersed throughout the country.

COORDINATING R&D BY SHARING OF DATA AND SOFTWARE

From the outset, an effort was made to promote the sharing of data and software development tools among the Initiative's research and development contractors, both to increase the focus on a common set of domain-specific problems, and to reduce the amount of effort put into redundant "tool building". The CPE Repository was established at the inception of the initiative to serve as a central clearinghouse for these tools and information. Housed on a UNIX workstation system and accessed by FTP, it has been used ever since as the primary means of collecting and disseminating data, sources for reusable development software tools, demonstration prototype software systems , and software collected or developed by the intiative's contractors.

One of the two major motivations for the CPE Repository was to help coordinate the many software development efforts that were to go on at different laboratories as part of the initiative. In anticipation of the issues that always arise when software systems developed in different places are to be integrated, some community "standards" were established for basic hardware and software underpinnings of the prototype software to be developed. These community standards covered such things as the type of UNIX workstations to be used, the version of Common LISP to be used, and also included the adoption of the Common Lisp Interface Manager for user interface development. Other standards were set for such things as X-windows, TCP/IP, etc.

On top of these basic necessities for software development, a range of software tools were added to the repository over time by BBN/ISX and other participating Planning Initiative contractors. These included such things as graphical interface tools (knowledge-base network browsers, a scientific graphics plotting package, a map display system, supplied by BBN), system development tools (a standard LISP DEFSYSTEM, logical pathnames), a knowledge-base maintenance system (**LOOM** from USC/Information Sciences Institute), a knowledge-directed database-access mechanism compatible with LOOM (**LIM/IDI** from UNISYS), a knowledge-based systems intercommunications protocol (**KQML** from UNISYS), and an object-oriented distributed communications mechanism that works with many different hardware platforms and programming languages (**CRONUS** from BBN)

Using these and other tools from their own laboratories, as appropriate to their research agendas, a number of other "technology packages" were developed by initiative contractors and installed in the CPE Repository together with documentation. Some contributions were primarily technology demonstrations, while others were intended for use directly in other development efforts. Among the contributions, there were several implementations each of generative planners (**SIPE** from SRI, **O-Plan** from the University of Edinburgh) , temporal information maintenance systems (**TMM** from Honeywell, **Tachyon** from GE Center for Research and Development), constraint-based scheduling systems (**OPIS** from CMU Robotics Institute), and intelligent database-access and database query planning systems (**SIMS** from USC/Information Sciences Institute, **CoBase** from UCLA). Other contributions included decision support and analysis tools (**DEMOS** and **DT** from Rockwell International's AI Laboratory), a knowledge-based systems development tool for reasoning with uncertainty (**PRIMO** from GE CRD) and a LISP-based statistics and metering package to support quantitative experiment analysis for AI systems (**CLIP/CLASP** from the University of Massachusetts).

Over the course of the initiative, a number of military planning applications were developed using these largely "domain-independent" AI tools that tailored or adapted them to the military transportation planning domain. Table 1 briefly describes a number of these. Many of these systems were developed for or in conjunction with IFDs or TIEs. All of the ones listed have all been incorporated into the CPE Testbed.

- **SOCAP** (SRI) An application of the SIPE-2 generative planning system to military employment planning, SOCAP consists primarily of a domain model and operators for a range of plans to address variants of the IFD-2 scenario. It also includes some additional tools for scenario data management and for communication with other Planning Initiative components. It also has its own user interface.
- **CAFS** (GE CRD) An application of Case-based Reasoning to the task of major force selection during employment planning, CAFS was designed to work in conjunction with SOCAP, or a human user. It is built on top of PRIMO and a Case-based reasoning system developed by GE called CARET, scheduled for release into the CPE this year.
- **DITOPS** (CMU Robotics Institute) A re-engineered version of the OPIS constraint-based scheduling system, with a constraint set and support tools to enable its use in military deployment transportation scheduling and transportation resource capacity analysis.
- **KTS** (Kestrel Institute) KTS is a very efficient constraint-based scheduling system for transportation deployment scheduling that was developed using Kestrel's KIDS semi-automatic program generation environment.
- **FORMAT** (MITRE) A knowledge acquisition tool for retroactive annotation of Force Modules, to enable their indexing and reuse, as by a case-based reasoning system (or a person).
- **LOGGEN** (MITRE) A tool for computing the amount of "sustainment" (food, supplies, etc.) that will be required by a military operation. LOGGEN has recently been incorporated into the DART system for use in operational contexts.
- **PFE** (BBN and CMU) The first tool developed specifically for the CPE, and one of the most frequently "checked out" of the CPE Repository, PFE is a LISP implementation of a military transportation simulator, modeled after the ones used in DART by the military to estimate the feasibility of a "TPFDD" deployment plan, given a set of available transportation resources. PFE's data input and output requirements were used as a model for both DITOPS and KTS.
- **FMERG** (BBN and ISX) FMERG was developed as a means of expanding SOCAP-generated plans into military "TPFDDs" or deployment plans for IFD-2. It is a simplified model of a force package elaboration and augmentation system. It consists of several subsystems that together span the gap between a high-level description of a set of forces with missions, to be deployed as part of a plan, and the detailed descriptions of those forces and their subcomponents, augmented by various kinds of combat support and sustainment elements, all of which need to be deployed.

Table 1: Prototype Planning Systems

The other important aspect of the CPE Repository was its role as a clearinghouse for domain knowledge and data about the military's transportation planning process. For this reason, the repository included a large body of textual materials detailing the processes and sample problems related to joint transportation planning derived from military officer's training courses on the subject, and data files containing a variety of sample datasets, taken primarily from unclassified military training materials and IFD demonstration scenarios.

THE KRSL LANGUAGE AND THE SHARED DOMAIN ONTOLOGY

In order to promote a consistent way of accessing all of the data and different kinds of representations of objects, actions, time relations, and plans, that are inputs or products of the military planning process, a substantial effort was undertaken during the second year of the project to develop a representation language specification that could be used uniformly to describe and store the domain data in the repository. This language, dubbed **KRSL** (pronounced just like 'carousel') for **Knowledge Representation Specification Language**, was a joint effort of many members of the initiative. In addition to basic object and concept descriptions, KRSL includes forms for units of measure, general relations and propositions, temporal relations, plan/goal descriptions, and producer-consumer constraints.

On top of this basic definitional syntax, substantial fragments of a uniform **Shared Domain Ontology (SDO)** were developed, describing all of the data elements and object types that participated as inputs or outputs to the systems that could be coupled in solving sample planning problems. This ontology is really a vocabulary of objects and relationships spanning all of the different kinds of transportation resources, ports, equipment, and other data structures that participate in the planning process. In its current form, it is sufficient to handle all of the information involved in the kinds of planning problems captured during the IFD-2 demonstration. More recent efforts to extend it have focused on extending the additional concepts involved in NEO operations, as in IFD-3, and in air campaign planning..

The CPE repository includes a large volume of data in its original textual and fixed-format ASCII forms. However, in order to make the data more uniformly available, a concerted effort was made to translate selected subsets of that data into KRSL using the concepts in the SDO, so that researchers and developers could draw on a consistently described reservoir of domain data, and appeal to a consistent vocabulary for that data. This standardization process began during the TIEs, and was fully realized in the CPE testbed. In the CPE Testbed, all communications between modules were TCP/IP based messages in terms of the KRSL language, and using the vocabulary provided by the SDO.

TECHNOLOGY INTEGRATION EXPERIMENTS

In the spring of 1992, following IFD-2, planning began for a series of Technology Integration Experiments (TIEs) that would develop and demonstrate interoperability between emerging initiative technologies, collect information about trade-offs between technical alternatives, and validate various technologies for potential use in IFDs. As a group, these TIEs demonstrated most of the elements of infrastructure and planning technology to be introduced into the Common Prototyping Environment over the succeeding 24 months.

The TIEs were explicitly designed to be experimental in nature: the intent being to explore methods of combining technical components that had been designed to play specific roles in planning or problem solving, and test designers' claims for their technology's applicability. When possible, we sought to contrast different mechanisms providing the same or similar functions, such as maintaining and reasoning with temporal information. For example, TMM and Tachyon, two systems for temporal constraint maintenance were made to interoperate with a plan generation system (SOCAP) during course of action generation (COAG). By sharing a standard interface, the two could be directly compared performing the same task. In other cases, TIES focused on exploring the potential of pairs of technologies that might mutually benefit from interoperation.

Each TIE included a 'Final Exam' in which the performance of the technical components involved was to be measured against some baseline criteria. The process of conducting TIE Final Exams was in part to identify and prepare component technologies for insertion into an IFD, and, in part, to benchmark performance on typical military planning and scheduling problems.

Table 2 lists the TIEs that were completed during 1992, all of which contributed to the development of the initial CPE Testbed. Figure 1 places these ties in the context of a simple "waterfall" model of the overall military plan development process. The top of the figure shows TIE #1, which was a first attempt to prototype several key infrastructure components; knowledge-directed information access and a knowledge-representation based distributed communications infrastructure. This TIE laid the groundwork for all of the components participating in the planning process to be able to rely on a uniform means of accessing and communicating information about the plans under construction. It introduced the notion of a "knowledge service", an intermediary between planning systems and data and knowledge bases.

The three major functional areas of plan development activity are also represented in the figure. Course of action generation (COAG) is the first, deployment plan expansion the second, and plan analysis the third. COAG involves strategic and tactical planning for both employment and deployment of forces. In the TIEs, SOCAP, the SIPE-based plan generator was the locus of most of the activity relating to COAG. Different TIEs examined how SOCAP could be improved, in terms primarily of the *quality of plans produced* by taking advantage of three different kinds of external functionality: Temporal constraint management, case-based reasoning for force selection, and constraint-based reasoning for transportation resource analysis. Both the use of temporal reasoners and constraint-based schedulers in this regard was an attempt to get early feedback to the COAG process from different kinds of automated *plan evaluation tools*

The second major focus in the TIEs was on Force Expansion, the process of producing a TPFDD or detailed deployment plan from high level inputs about major force groups to be deployed (the output of the COAG process). TIE #3, described below, used constraint-based scheduling techniques to develop reasonable deadlines and choices for mode of transport (air, sea) for the forces during deployment planning.

THE INFRASTRUCTURE TIE: DISTRIBUTED KNOWLEDGE ACCESS

The objective of the Infrastructure TIE was to demonstrate integration and interoperability technology for the Common Prototyping Environment and new Planning Initiative software generally. It was developed with the premise that knowledge-based software systems could interoperate most effectively with other software systems in a distributed environment using a consistent representation language as 'interlingua'. They could thereby avoid having to develop and maintain independent means of translating messages between every pair of communicating systems. The TIE demonstrated the feasibility of this integrated approach to inter-agent communications and the sharing of knowledge among knowledge-based software agents and knowledge-level access to common databases. It was a joint project with participants from BBN, ISX, SRI and Paramax (previously and now Unisys). It formed the basis for the first working prototype of the CPE infrastructure.

1. **Infrastructure TIE (BBN, ISX, Unisys)** demonstrated the knowledge services and distributed, knowledge-based communications elements of the envisioned distributed CPE planning environment.
2. **Case Based Reasoning - Generative Planning TIE (GE-CRD, SRI)** applied a CBR system developed at GE to do operator role selection for force units used in a military operations plan built by a generative planner.
3. **Constraint Based Scheduling - Deployment Planning TIE (CMU, BBN, SRI)** demonstrated constraint-based scheduling system in a heterogeneous military planning system. **TIE #3A** used deployment constraints to do the final stages of deployment plan development, producing a feasible TPFDD. **TIE #3B** used the scheduler as a means of doing preliminary deployment plan analysis during the intial phases of COA development, as a means of filtering the options for an operations plan.
4. **Temporal Reasoning - Generative Planning TIE (GE-CRD, Honeywell, SRI)** used a temporal reasoning system (Tachyon or TMM) during generative planning to propagate the temporal constraints in a developing plan. This arrangement enabled some comparative experimentation with Tachyon and TMM.
5. **Temporal Reasoning - Case-based Force Expansion (GE CRD, BBN)** applied temporal constraint management during case-based force selection and expansion, where selected forces are "unpacked" to form elements of a TPFDD.
6. **Case-based reasoning - Force Expansion (GE CRD, BBN, MITRE)** used CBR techniques to refining and enumerating the elements of forces selected during high-level planning.

Table 2: 1992 TIEs

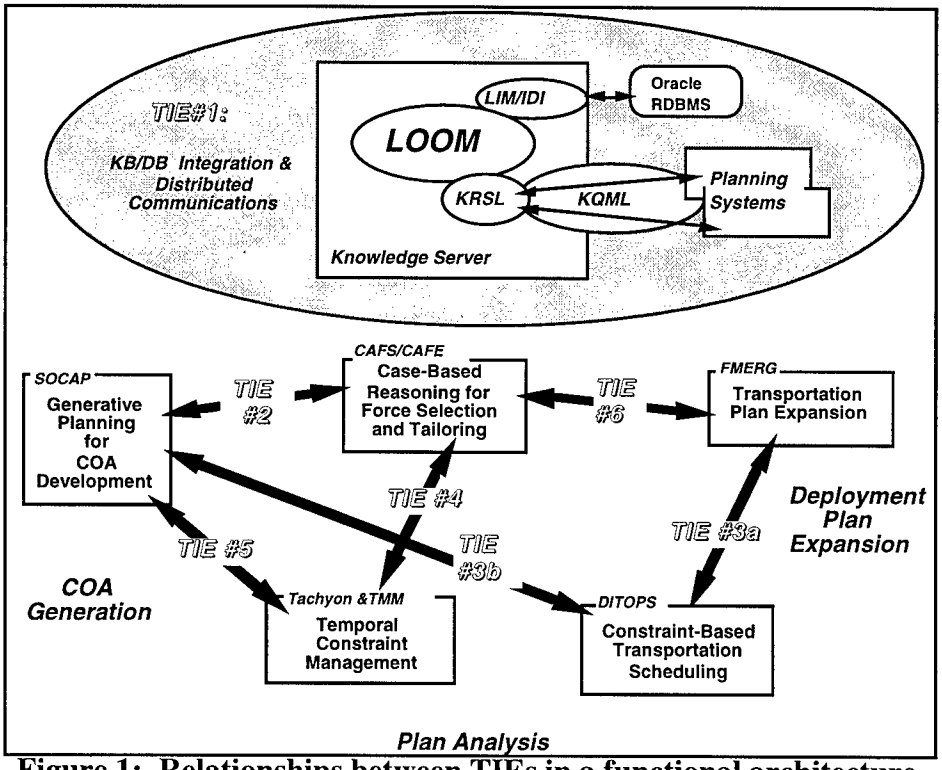

Figure 1: Relationships between TIEs in a functional architecture.

The language used as interlingua in this experiment, KRSL, was developed for the CPE by ISX, in collaboration with BBN and many members of the initiative community, who contributed their expertise on the aspects of the language that they were most concerned with. The language provided a simple syntax for definitions of concepts and instances, relations, units, and some of the basic elements of an ontology for representing plans, specifically temporal relations and events. An implementation of the KRSL language, the 'KRSL Kernel' was developed as a software layer that translated forms, queries and assertions into and out of the LOOM knowledge representation system, developed at USC/Information Sciences Institute (MacGregor and Yen, 1989; MacGregor and Burstein, 1991). To construct a 'knowledge server' to play the role of information broker in the distributed architecture, LOOM was also integrated with Paramax' Intelligent Database Interface (IDI) through an intermediate layer called LIM (LOOM Interface Module). This provided access to a set of persistent relational databases maintained in Oracle's (tm) RDBMS.

The generall architecture of the knowledge server and communications substrates that were developed during the Infrastucture TIE and used later in the CPE is represented in Figure 2. The TIE communications software was built around Paramax' implementation of KQML, a distributed communications model for intelligent software-agents developed from a specification that was the work of a number of AI researchers under the auspices of the DARPA Knowledge Sharing Project. KQML supported the message encapsulation and routing of messages composed of KRSL expressions. During the TIE, messages were passed among the Knowledge Server, FMERG and PFE, the latter two being components developed by BBN, ISX and CMU that had participated in the previous IFD-2 demonstration where communications were by file transfer or by pointer in a shared LISP data space.

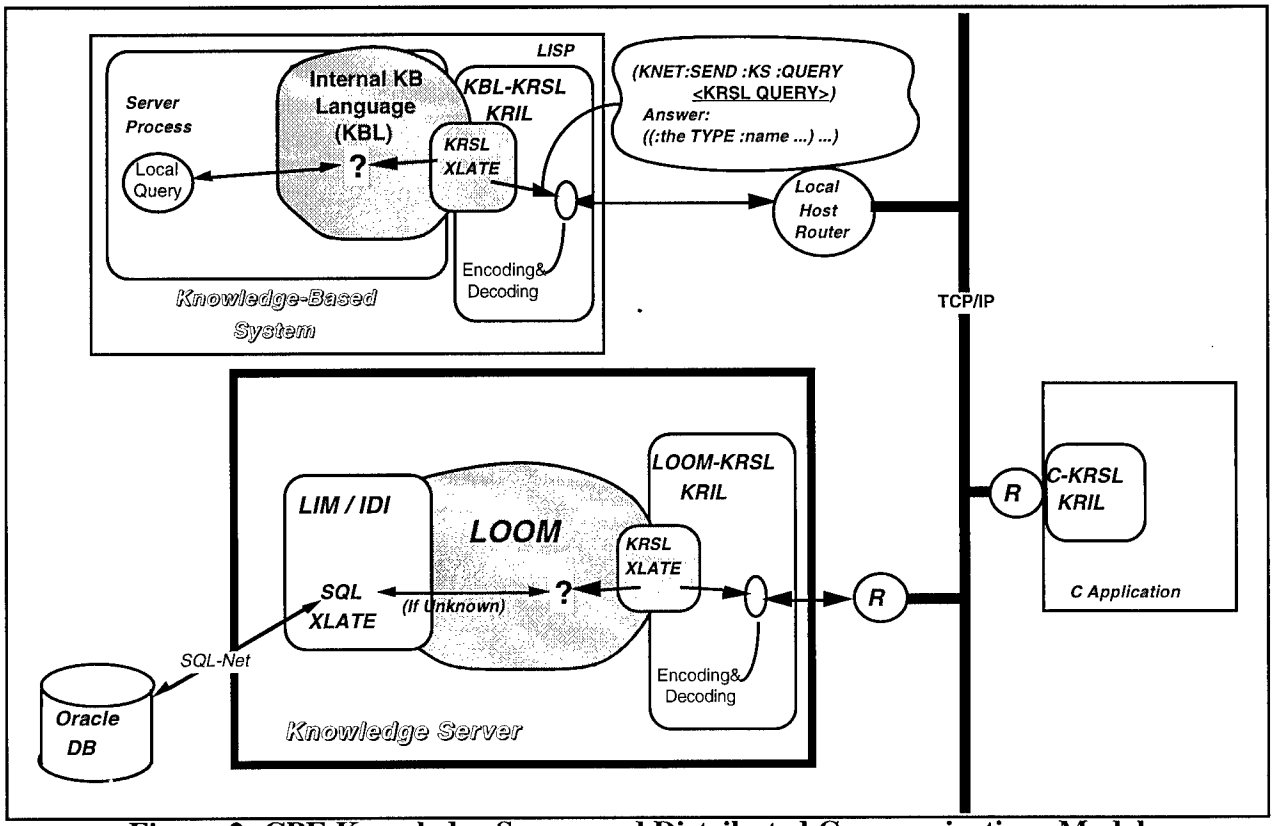

Figure 2: CPE Knowledge Server and Distributed Communications Model

The KQML communications mechanism was implemented as a "wrapper" consisting of two pieces of software that were added to each component "agent" that was to be capable of remote communications. The two pieces are a "Router" and a "Knowledge Router Interface Library" (KRIL). The router is a program that formats expressions in KQML message syntax and forwards them through the distributed network. There is one router per implementation language (e.g., there is one Common Lisp router). The KRIL is a *representation language specific* interface to the router that acts primarily as a representation translator (e.g., from LOOM to KRSL and back) and local dispatcher for the set of KQML dialogue primitives (e.g., queries, replies and assertions) that will be supported by the agent. In the TIE, FMERG, PFE and the Knowledge Server (KS) all used the same Common Lisp router; FMERG, and the KS used the KRSL KRIL to translate KRSL messages into their local LOOM representations; PFE used a special purpose KRIL that later was developed into a more general package to support the translation of KRSL into Common Lisp Object Specification (CLOS) descriptions.

Communications between these agents, consisting primarily of representations of military plans and deployment schedules were standardized as much as possible using what then existed of an evolving Shared Domain Ontology (SDO), a common set of terms, concepts and relations defined in KRSL to be used for all communications between agents in the CPE environment. This catalog of definitions included standard frames or concepts for such things as air ports, sea ports, geographic locations (called GEOLOCs), transportation movement (TPFDD) plan elements, and specific representations required to enable communication among the modules involved, e.g., the major force list representation of SOCAP plans and the representation of force modules developed from IFD-2.

The Infrastructure TIE also developed the first implementation of a Knowledge Server (KS), an agent whose primary purpose was to act as a maintainer of shared knowledge and information, and a mediator to databases containing additional information. The Knowledge Server consisted of the KRSL Kernel, LOOM, the Intelligent Database Interface (IDI), and the LIM interface between LOOM and IDI. IDI also communicated with an external Oracle data base containing most of the relevant data from IFD-2, including geolocs, force descriptions, etc. The Knowledge Server supported external queries and assertions in the KRSL interlingua, communicated via KQML. The information in the answers it provided could either come directly from the LOOM knowledge base or from the Oracle data base, via LIM and IDI.

The final demonstration for the TIE consisted of recreating a portion of the IFD-2 demonstration, this time using KRSL and KQML to access and store intermediate information and results in the Knowledge Server. A SOCAP major force plan was asserted into the knowledge server using KRSL. FMERG was invoked, and issued KRSL queries to the knowledge server for the major force plan and definitions of the forces involved, and received that information back in KRSL forms based on the SDO definitions of the objects involved. It then produced a TPFDD for that plan, and asserted that result back into the Knowledge Server, from which it could be retrieved by PFE for a transportation feasibility analysis[2]. The overall Infrastructure Prototype configuration is extremely flexible, allowing various components to reside on the same workstation or be distributed across the Internet. The specific configuration demonstrated had FMERG running at Rome Laboratory, communicating over the internet with a Knowledge Server running at Paramax in Philadelphia.

---

[2]This last step, connecting the Knowledge Server to PFE, was accomplished a short time after the initial TIE was completed.

Much of the initial infrastructure was instrumented to collect timing information. As might be expected for a first prototype, there were many performance issues outstanding, and some steps were quite slow, especially given the size of the data that was transmitted (e.g., 20 minutes to retrieve approximately 1MB of information about forces for a plan). This information served as a guide to us in the subsequent development of the CPE Testbed.

The attempted integration of KRSL, LIM and LOOM also raised some representational issues and inconsistencies that needed to be addressed. One of the more interesting of these was the issue of how to handle the relationship between class-level information as stored in database records and the corresponding classes or concepts in the knowledge base. For example, a data table might contain information about types of aircraft (e.g., their average speed, cargo capacity, etc.) which might be represented as attributes of class-level concepts in LOOM or KRSL. Unfortunately, the versions of LIM and IDI at that time only supported the mapping of database records into instances. Another issue was the mapping between KRSL objects with subparts and representations of that information in the data base. Recursive or cyclic relations are well known to be difficult for relational databases to handle effectively.

USING EXTERNAL REASONING TOOLS IN SUPPORT OF GENERATIVE PLANNING

In tackling a real-world problem using a single, uniform strategy AI reasoning tool, it is not uncommon to find that the real-world problem requires solving subproblems which are best approached using different strategies. In some cases, legacy software must be integrated with the AI system; or the AI system must be altered to fit the new problem (e.g., with a customized user interface); or additional automated reasoning techniques must be added to round out the capabilities of the AI system for work in that domain. Several of the TIEs provided some unique experience with supplementing a mature generative planning tool with several other substantial pieces of AI technology: a case-based reasoner; a temporal reasoner, and a scheduling system. These technology integration experiments were among the first examples where mature AI systems whose development paths were completely orthogonal were harnessed to solve the same problem.

The goal of these integration experiments was to improve the quality of the plans generated by SOCAP. One TIE (#4) experimentally combined temporal constraint propagation systems (Tachyon, TMM) with SOCAP to improve SOCAP's temporal reasoning capabilities. This integration resulted in SOCAP being able to represent more sophisticated temporal constraints within plans and to reason more accurately about the times and durations of actions and about resource utilization over time. Another TIE (#2) incorporated a case-based reasoning (CBR) system, CAFS (for Case-based Force Selection) to extend SOCAP's ability to choose objects to participate in operations. CAFS selected appropriate forces from a library based on similarity comparisons between those forces and the operator requirements and context provided by the new plan. A third TIE (#3B) integrated SOCAP's planning capabilities with DITOPS (Distributed Transportation Scheduling in OPIS), a constraint-based scheduling system developed at Carnegie Mellon University. SOCAP's ability to generate robust, feasible plans with realistic allocation of resources was improved by using feedback from the scheduling system directly during planning.

These integration projects were unique in two ways: first, they utilized existing, independently developed AI-based modules to supplement an existing generative planning system; second, they add capabilities that are novel or relatively unexplored in generative planning systems.

TEMPORAL REASONING IN SUPPORT OF GENERATIVE PLANNING WITH SIPE-2

When SOCAP was originally developed as an application of SIPE-2, SIPE-2's limited temporal reasoning capability surfaced as a shortcoming. Before the TIE, SIPE-2 was unable to reason about utilization of resources and could not place temporal constraints between actions in the plans. Consequently, the plans generated did not represent important constraints that existed in the planning application domain.

In previous versions of SIPE-2, time was treated strictly as a consumable resource; that is, it could be consumed but not produced, and its consumption over parallel tasks was nonadditive. Each action specification could have associated start-time and duration variables, but SIPE-2 calculated specific values for time variables only when the constraints forced a particular value; otherwise, the allowable range was computed.

SIPE-2 has several techniques for establishing the relative orderings of actions: inserting ordering links to avoid resource conflicts; using one action to meet several other actions' precondition requirements by ordering those actions after the action that achieved those conditions; and coordinating separate subplans by adding ordering links between subgoals of the two subplans. These capabilities allow many simple temporal problems to be solved, but the inability to represent the time constraints of two possibly unordered actions was seen as a problem. Two SIPE-2 actions are either ordered with respect to each other, or they are unordered. If the latter, the planner considers it possible to order them either way or to execute them simultaneously. It was not possible to model *when* the various effects of an action become true during its execution, or that must occur *simultaneously*. By adding as a support system a temporal constraint manager for Allen's (Allen, 1981) temporal relations calculus, it became possible to explicitly represent actions starting or finishing at the same time, overlapping each other, or one occurring during another. Many dependencies between different military actions should be represented in this way in SOCAP. For instance, cargo off-load teams should arrive at the same time as the first air or sea transport arrives at the destination airport or seaport.

For the TIE, SOCAP's ability to represent and reason about time was extended by adding a layer on top of SIPE-2 that would keep track of the temporal constraints within the plan, initially using the Tachyon temporal reasoning system (Arthur, Deitsch and Stillman, 1993) to maintain and propagate these constraints. The interface to Tachyon was designed to be general enough to permit a different temporal reasoner to be substituted (in particular, TMM (Dean & McDermott, 1988; Schrag et al. 1992).

Our approach to the integration of a temporal reasoning engine was somewhat "coarse-grained". A new plan critic was written for SIPE-2 to be run at the end of each "planning level", rather than as each new constraint was encountered during planning. This critic extracts all of the temporal information (time windows and internode constraints) from the plan, sends it to a Temporal Reasoner, and stores the updated time windows returned back into the nodes in the plan. In addition, methods were added to maintain these constraints on the plan as goals are expanded to a new level by SOCAP. SIPE-2's operator syntax was extended to allow a designer to specify any of the 13 Allen relations or quantitative constraints (the permissible range of metric distances) between the endpoints of any pair of nodes in an operator.

The extended system found temporal inconsistencies in previously generated plans which could only be resolved by changing the dates on which military units were available to perform missions, or by assigning different units to those missions. This shows that the system now reasons with a more complete model of the constraints operating in the military domain. The temporal information is especially important for proper use of the planner's products by

downstream scheduling software, since SOCAP is able to pass a more complete and consistent set of constraints to the scheduler. Although the temporal reasoning introduced into SIPE-2 by the TIE cannot be considered "complete" since representation and reasoning support for both continuous and interruptable events are still lacking, the limited capabilities that were added provide a significant source of power in dealing with important constraints in the domain of military planning and improve the generated plans substantially.

CASE-BASED REASONING FOR FORCE SELECTION IN PLANNING

Selecting the right force to participate in a military operation and tailoring a force to meet special requirements of a specific operation are two important parts of operations planning. Considerations such as a unit's potential to deter or defend against an enemy threat, its mobilization, its ability to handle the terrain, and its time to deploy must be reviewed. When it was initially developed, SOCAP always asked the user to select the force units that would be used to achieve specific missions, by presenting a list of available units that met the constraints of the operator (to perform the mission) when expanding a goal for which that operator applied. The user could see what constraints were met by the units in the list but had to rely on personal knowledge of the mission context, and capabilities of the each possible unit to determine which was most appropriate in that context. SOCAP itself had no way of ranking or preferring some units over others if they all satisfied the *necessary* constraints on their utilization for a given job.

This kind of preferential force selection and tailoring was seen as an area where case-based reasoning (Kolodner, 1993) would apply. TIE #2 was an addressed this issue by integrating a CBR system from GE's Center for Research and Development called CAFS (CAse-based Force Selection) which was built on GE's CAse-based REasoning Tool (CARET). In CAFS, descriptions of forces from a case library of force units are indexed and retrieved based on their mission requirements, climate, terrain, mobility and other related information.

For TIE #2, SOCAP was modified to call the CAFS module for major force selection instead of presenting a list to the user for selection. CAFS uses the constraints and context information provided by the SOCAP operator and bindings to find and rank potential force units as to their appropriateness for a given mission (including ones that were tailored to specific missions in the past). If the closest matching force does not fit SOCAP's requirements, heuristics can be applied within CAFS to modify the force in the appropriate manner. For example, it might be necessary to add additional support units to the retrieved force structure to account for differences between the prior and current situation.

CONSTRAINT-BASED SCHEDULING AS A CAPACITY ANALYSIS TOOL DURING PLANNING

An important feature of the type of decision support that SOCAP is intended to provide is a capability for replanning based on feedback from (usually external) plan feasibility evaluation tools. Assessing the transportation feasibility (essentially the use of resources over time) of a plan is a major point of focus for military planners. To investigate this issue, and also explore ways of overcoming SOCAP's simplified model of resource management, TIE #3 explored an integration of SOCAP with CMU's DITOPS system for constraint-based resource reasoning and scheduling. In this TIE (#3B), SOCAP was modified to call DITOPS at various stages of its search through the space of possible plans to assess the feasibility of the current partial plan from the standpoint of transportation resource capacity requirements. This early analysis aids in the assignment of resources to operations based on projections of resource bottlenecks. That is,

either SOCAP or a user can use the analysis results to choose feasible deployment destinations for major forces during initial plan generation or to reassign transportation resources.

To focus DITOPS' attention on the relevant parts of the plan, SOCAP extracts a temporally ordered plan network that included only the transport operations and other non-resource-using actions that contributed temporal constraint information. For each operation, it supplies the transportation resources (planes, ships) assigned to transport specific units, and holding capacity required on those resources to transport the personnel and materials involved. This information is analyzed by DITOPS' capacity analysis routines and the result is passed back to SOCAP in the form of comparative estimates for the expected availability versus demand for aggregate resources, by class of transport, over time. This data is presented on a color graph showing expected demand on a resource type vs. available capacity over time. Users (and, eventually, SOCAP itself) can then reassign resources or change the time of an operation.

FORCE ELABORATION AND THE TRANSPORTATION SCHEDULING TIES

A TPFDD (Time-Phased Force Deployment Database) is the current-day operational data form of a military deployment plan. It is a table of all of the individual units to be deployed by air or sea to specific destinations, their sizes (in tons of air or sea cargo), and the dates that they are available for shipment and need to arrive at their destinations. TPFDDs are used as inputs to simulators to estimate the transportation feasibility of a plan in plan analysis tools such as DART. They are also input to scheduling algorithms used to produce detailed schedules and manifests for shipments of air and sea cargo. They are thus a critical link in the data path for current military planning operations.

The current-day process of developing a TPFDD is a labor-intensive function, primarily performed by the command staffs of individual services (Air Force, Army, Navy). Once a general operations plan has been developed by the Joint Staff, the services are tasked to "fill in the details" of the operations, and determine the lists personnel and equipment that will be needed. They often do this by of "cut and paste" operations on detailed force lists developed for prior missions, although this is largely a paper and pencil and "fat finger" data entry task without the proper software support tools. Since TPFDDs can number in the thousands or tens of thousands of records for large operations, this is a labor-intensive task, indeed.

In order to fill in required data elements in each TPFDD record, these service planners must attempt, sometimes without all of the necessary information, to do a rough *scheduling* of the deployment of those units, specifying the ports and the time windows for the arrival and departure of each unit from ports along the path of their deployment to a theater of operations. Frequently, they do this with only approximate information about the transportation resources that will be available, and even less information about the constraints on loading and unloading of cargo at individual ports.

They are forced to do this because they have no way of *specifying directly what they do know,* namely the ordering and timing constraints (e.g., which units need to arrive before other units) that are based on their knowledge of the plan under construction and military planning more generally. If they were able to specify this information directly and conveniently, then more dynamic and automated means of scheduling and rescheduling the transportation elements of the plan would be possible when better information was available.

Within the Planning Initiative, we have sought to demonstrate a path to automation of some of these staff functions, at least to the point of demonstrating how automated tools could help, given the right information. In IFD-2, BBN and ISX developed FMERG as a very simplified first step

toward automated TPFDD generation directly from operations plans produced by SOCAP. FMERG approximated parts of this currently manual process, expanding each force mentioned in the SOCAP plan into a list of units and materials by using a force module library, augmenting the list of operational units to be deployed with a number of other units serving combat support and service support functions (local transportation, medical, food services, etc.), and, finally, adding units to the deployment plan based on doctrinal rules about resupply for things like food, fuel and ammunition (this last activity is more fully realized in the system LOGGEN, developed by MITRE).

FMERG was designed to be replaced by technologies developed by other Planning Initiative R&D projects. Two of the TIEs were designed as steps in that direction. In one, still under way, GE-CRD has been working on extending their Case-Based Reasoning tool, CAFS, to provide more capability to do intelligent tailoring of those configured units to adapt them to new situations. In the other TIE, CMU developed scheduled departure and arrival time information for TPFDDs using the DITOPS constraint-based transportation scheduling tool.

CONSTRAINT-BASED TPFDD SCHEDULING

DITOPS is a constraint-based scheduling system for military transportation planning based on the OPIS system for job-shop scheduling (Smith, 1994). In the job-shop scheduling domain, there are typically a number of ways in which manufacturing of an item might be accomplished, by using sequences of operations, each of which can be accomplished by using tools that exist at multiple stations on the factory floor. In transferring this model to transportation scheduling, DITOPS was made to work with largely completed TPFDDs, which specified the goods to move, and required destinations, but could leave open some of the decisions about *which resources* would be used to move the goods, and, potentially, *which intermediate ports* the goods would travel through. DITOPS then builds detailed schedules of which ships and planes will be used to transport each good, in effect combining a transportation feasibility analysis similar in spirit to what military simulators had done before with the development of actual schedules.

TIE#3 experimented with relaxing some of the constraints on how complete a TPFDD was required to do this kind of scheduling, in effect opening up the possibility that DITOPS would take over some of the decisionmaking relating to port and mode (air or sea-based transport) assignment, rather than taking the overly restrictive prescriptions embodied in a TPFDD, and just scheduling air and sea craft to carry cargo along proscribed routes. These kinds of choices are the analog in the transportation planning problem of the choice in the job-shop world about which of several paths through the factory can be used to complete a manufacturing task in the most timely fashion.

For these experiments, DITOPS was made to use a less constrained description of the plan than that normally embodied in a TPFDD, leaving open choices to be made in terms of which ports ships could use as destinations for each load, and which kinds of transportation resources to use (air or sea craft) to get the most effective transportation plan. Normally, the military uses a rule of thumb that, for large operations, 20 percent of the cargo should move by air, and 80 percent by sea. However, given timing considerations and available lift options, this heuristic can easily be wrong. By relaxing some of these constraints on the shape of the scheduler's output, but providing more information about the structure and dependencies in the plan, more efficient plans and schedules can often be produced, and they are certainly more readily revised.

To achieve this, DITOPS was provided some additional information about the doctrinal constraints on which elements of a force needed to move before other elements, replacing fixed,

but estimated, dates for unit movements found in a TPFDD. Given this additional information, DITOPS was able to demonstrate the potential for automating some mode and port decisions based on resource capacity information. DITOPS was able to balances the use of ports, and avoid some bottlenecks due to port capacity overload. Similarly, by avoiding early commitments on how cargo was to be shipped (air or sea modes) the scheduler was able to find efficient means of transportation for each unit (for example, using excess air transport capacity to improve arrival times) and better balance the limited resources available. This TIE successfully demonstrated how a more appropriate encoding of the available information could lead to better, and more easily and automatically revisable, transportation plans.

THE CPE TESTBED: AN EXPERIMENTAL DISTRIBUTED PLANNING ENVIRONMENT

Tying the TIEs Together

After the initial round of TIEs was completed, work began in earnest to bring together all of the technology components involved into a single distributed environment called the CPE Testbed, based on the interoperability model used in the Infrastructure TIE. Beyond this unifying function, the CPE Testbed was designed to:

- Provide a support infrastructure for future TIEs that included support for experimental metrics and their evaluation and uniform access to data and scenarios for the transportation planning domain.
- Establish a common representational basis for all remote communications between planning and scheduling systems.
- Facilitate the transitioning of technologies into IFDs and operational prototypes.
- Enable exploration of issues of coordination and control in a heterogeneous system for replanning and rescheduling.

CPE Testbed Release 1.0 was developed from the systems represented by the TIEs in an intense five month effort involving all of the TIE participants during the spring of 1993. During that time, all of the ad-hoc communications that has occurred in the TIEs were transformed into consistent KRSL representations, including the development of KRSL plan representations and a translator from SIPE-2 plan representations to KRSL forms. Support for remote starting and metering of software modules was developed, in part by replacing some of the underpinnings of the prototype KQML software with a more mature object-oriented communications substrate[3] and by adding a simple, uniform interface to that functionality as a package called KNET.

A substantial amount of effort and voluminous EMAIL traffic was devoted to standardizing the language of communications between the modules using KRSL. To standardize the functionalities of the different systems for experimental comparison purposes, all communications between modules were organized around standard sets of messages to be defined for pairs of functional classes of software modules. Systems were typed as Plan Generators, Force Selectors, Temporal Reasoners, Force Expanders, Deployment Simulators/Schedulersn and Knowledge Servers. For each type of module, KNET messages that it would have a contract to respond to were defined, inputs and outputs characterized

[3]BBN's CRONUS system was used for this, primarily because of its relative maturity at the time, and its capabilities for multiple language support (including C, C++, LISP), and its tools for remote process control.

descriptively and in terms of KRSL syntax. Figure 3 shows the interactions paths for which these communications were defined.

At the time of the first CPE release, there was one Plan Generator (SOCAP) two temporal reasoners (Tachyon, TMM), one Force Selector (CAFS), one Force Expander (FMERG), two Deployment Simulators/Schedulers (PFE, KTS) and one Knowledge Server, plus a "client" system called the CPE Interface which was used for remote process and experiment control, but was not a server that responded to messages. DITOPS was soon added as a third scheduler/simulator. Recently, the communications between DITOPS and SOCAP (from TIE #3B) was implemented in the CPE as well, by giving DITOPS a second role as a Capacity Analyzer.

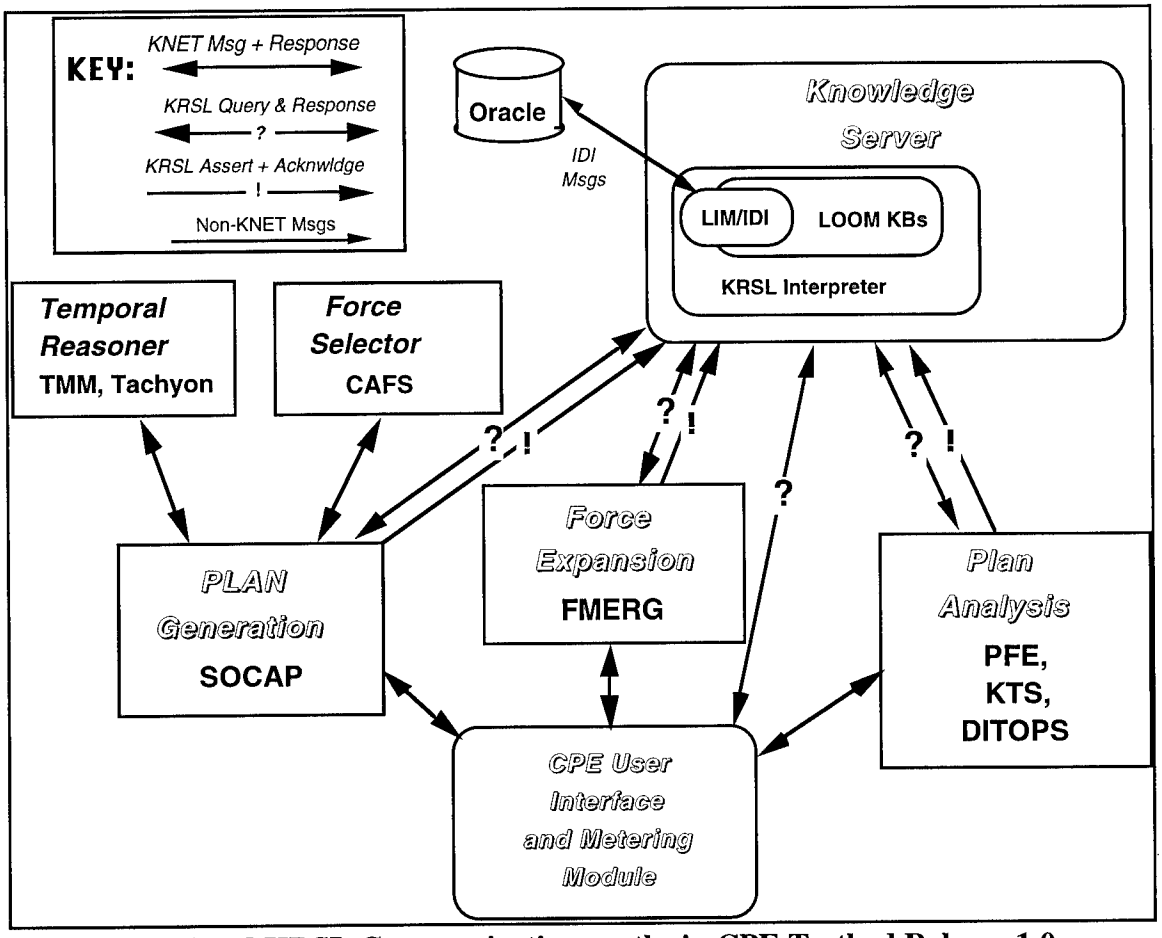

Figure 3: Defined KRSL Communications paths in CPE Testbed Release 1.0

The effort to standardize the content of communications between modules was largely successful, though time constraints forced some hard choices on the process. In particular, the communications between plan generators and the temporal reasoners continued to have the batch-oriented flavor of the original TIE interface between SOCAP and Tachyon, even though TMM, the other temporal reasoner, was designed for more incremental interactions with clients[4]. Similarly, there were some aspects of SIPE's plan language that are unique to that system which

[4]An experiment where TMM is used more as it was intended, in conjunction with the O-Plan planning is planned for the near future. This should lead to a revised interaction style for temporal reasoners in the CPE.

carried over into KRSL and the SDO. Ironing out some of these issues will require further effort by the research community as a whole.

Other improvements over the system represented by the Infrastructure TIE included an improved message encoding scheme that achieved a three-fold reduction in the size of large messages, and a ten-fold decrease in transmission time of those messages. This meant that the communication of a TPFDD object of transmission size over 1.5 MBytes during the TIE was reduced to .5Mbytes and the time reduced from 20 minutes to 2 minutes. Additional speedups occurred in accessing remote databases from the Knowledge Server as well.

CPE TESTBED AS A PLATFORM FOR TECHNOLOGY EVALUATION EXPERIMENTS.

It was a critical goal for the CPE Testbed that it be capable of and easy to run experiments that tested the speed and effectiveness of new technologies that were introduced into the environment, both singly and in TIE-like combinations. To this end, the process of running programs in the CPE was organized around the notions of an experiment with multiple trials, and a mechanism for adding 'alligator clips' or **meters** at various points in the computations of modules was included in the CPE and KNET software. By simple declarations, information about run times and process parameters could be collected on a trial-by-trial basis, where each trial might, for example, vary which module was performing a particular problem solving function. For example, Tachyon and TMM could be compared in sets of trials where they were called by SOCAP with the same sequence of inputs, simply by setting up the sequence of trials in the CPE Testbed Experimenter's User Interface. To evaluate the results of experiments, the CLASP analytical statistics package from the university of Massachusetts was incorporated into the CPE User Interface as well, so that data collected by CPE meters could be directly analyzed using a variety of statistical measures.

Also from the CPE User Interface, one could specify which hosts the modules would run on during a trial. All active hosts in the same CPE configuration, be they at BBN in Cambridge, ISX in Los Angeles, Rome Laboratory in New York state, or one of the developer sites at other locations around the country could be used during a single trial, if the modules involved were installed at those sites.

At the time of this writing, there are twelve modules different "modules" or "agents" that can communicate via the CPE's distributed communications model. These systems represent the contributions of ten different corporations and university laboratories, and several others will be added shortly. They include, in addition to the "CPE User Interface" module, SOCAP, Tachyon, TMM, CAFS, DITOPS, KTS, FMERG, PFE, FORMAT, the Knowledge Server, and an experimental "Plan Server" for plans developed by the TARGET plan authoring system from IFD-3. There are also a number of research prototype systems that are under consideration for future incorporation into the CPE, including OPLAN-2, SIMS, CoBase, and several others. Other, more operationally oriented systems under consideration include LOGGEN, AMP, a successor to DART, and ACPT, a plan authoring tool for air campaign planning.

SUMMARY AND CONCLUSION

The CPE, including both its testbed and repository aspects, has successfully demonstrated the potential for increased collaboration and sharing among a large set of research and development projects. It has served as a validation of the concept and methodology for promoting more effective software and information sharing, moved a large community of researchers toward mechanisms for validating the products of their work both in terms of increased effort to

demonstrate interoperability and increased attention to the establishment of experimental metrics for progress.

There are still many directions in which this environment could improve and mature. Some in the university research community have found the CPE to be less directly useful to them than we had hoped, partly because of when it became available, and partly because of its size and the compromises involved in designing an environment that integrates a wide group of software systems with current technologies. A number of other the problems that have surfaced and are being addressed at the current time. Most of the systems that have to-date been incorporated into the testbed were the applications of AI technologies that pre-dated the Planning Initiative. Addressing the needs of the university researchers developing the next generation of tools is now one of our biggest concerns.

Devising an architecture for a diverse, heterogeneous population of software systems all participating in solving complex planning problems is a challenging task. It is in some ways even more challenging when one understands that there will always be a strong human element to these distributed systems. The challenge we now face is not just to build knowledge-intensive distributred software systems, but systems in which many people, at many different locations are assisted by intelligent planning software in their dynamic, ongoing efforts to coordinate the activities of one of the largest single organizations in the world.

Acknowledgements. The portion of this work performed by BBN and ISX was funded by ARPA and monitored by Rome Laboratory under Contract F30602-91-C0014.

REFERENCES

Allen, J.F. "Maintaining knowledge about temporal intervals." *Communications of the ACM*, 26(11):832-843, 1983.

Arthur, R., Deitsch, A., and Stillman, J. "Tachyon: A constraint-based temporal reasoning model and its implementation." *SIGART Bulletin*, 4:3, July 1993.

Bienkowski, M.A, Desimone, R.V., and desJardins, M.E., "Decision Support for Transportation Planning in Joint COA Development: Annual Report." SRI International Technical Report ITAD-2062-AR-93-101. March 1993.

Dean, T.L. and McDermott, D.V., " Temporal Data Base Management, *Artificial Intelligence* 36:1-55. 1988.

MacGregor, R. and Yen, J. *The Knowledge Representation Project at ISI.* ISI Research Reports ISI/RS-88-199, July 1989, University of Southern California.

MacGregor, R. and Burstein, M.H., Using a Description Classifier to Enhance Knowledge Representation. *IEEE Expert,* 6:3, June 1991, pp. 41-46.

Wilkins, D.E., *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann Publishers, Inc., San Mateo, CA. August 1988.

Wilkins, D.E., and Desimone R.V., "Applying an AI Planner to Military Operations Planning." In M. Fox and M. Zweben, eds., *Intelligent Scheduling.* Morgan Kaufmann Publishers, Inc., San Mateo, CA. 1993.

Schrag, B., Carciofini, J., and Boddy, M. "b-TMM Manual (version b19)." Technical Report CS-R92-012, Honeywell Systems & Research Center, 1992.

Allen J. F. et al. *Reasoning About Plans* Morgan Kaufmann Publishers, Inc., San Mateo, CA. 1991.

Kolodner, J. *Case-Based Reasoning.* Morgan Kaufmann Publishers, Inc., San Mateo, CA. 1993.

Smith, S.F., "OPIS: A Methodology and Architecture for Reactive Scheduling", in (Eds.) Fox, M. and Zweben, M. *Intelligent Scheduling*, Morgan Kaufman, Palo Alto, 1994.

APPENDIX C:

ITAS:

A Portable, Interactive Transportation Scheduling Tool
Using a Search Engine Generated from Formal Specifications

Proceedings of the Third AI Planning Systems Conference,
Edinburgh, Scotland, The AAAI Press, 1996.

Proceedings of the Third International Conference on Artificial Intelligence Planning System (AIPS-96)

ITAS:

A Mixed-Initiative Transportation Scheduler Generated from Formal Specifications

Mark H. Burstein
BBN Systems and Technologies
10 Moulton St.
Cambridge, MA 02138
Email: burstein@bbn.com

and

Douglas R. Smith
Kestrel Institute
3260 Hillview Avenue
Palo Alto, California 94304
Email: smith@kestrel.edu

ABSTRACT

In a joint project, BBN and Kestrel Institute have developed a prototype of a mixed-initiative scheduling system called ITAS (In-Theater Airlift Scheduler) for the U.S. Air Force, Pacific Command. The system was built in large part using the KIDS (Kestrel Interactive Development System) program synthesis tool. In previous work for the ARPA/Rome Laboratory Planning Initiative (ARPI), Kestrel has used their *program transformation* technology to derive extremely fast and accurate transportation schedulers from formal specifications, as much as several orders of magnitude faster than currently deployed systems. The development process can produce highly efficient code along with a proof of the code's correctness.

This paper describes the current prototype ITAS system and its scheduling algorithm, as a concrete example of a generated scheduling working on a real problem. We outline the generated search algorithm in order to promote and facilitate comparison with other constraint-based scheduling systems. The overall system includes a database and interactive interface that allows users to control shape of the schedule produced in a number of ways. ITAS runs on a Macintosh Powerbook™ notebook computer, for reasons of portability.

INTRODUCTION

This report describes a prototype application of Kestrel Institute's research on the transformational development of high-performance transportation schedulers[5]. The system,

[5] This research was supported by ARPA/Rome Laboratories under Contracts F30602-91-C-014 (BBN) and F30602-91-C-0043 (Kestrel).

ITAS, for In-Theater Transportation Scheduler, is designed to assist human schedulers of airplanes who work to produce daily flight plans to move cargo and people within a specific theater of operations. For example, after the Iniki Hurricane in the Hawaiian Islands a few years ago, these airlift schedulers sent planes from Honolulu and other islands to the Hawaiian island of Kauai to deliver supplies and relief workers to that area. This scenario is not atypical, and was used as a working example as we developed the system.

Previously, Kestrel had demonstrated the potential to produce extremely fast and accurate transportation schedulers from formal specifications using the Kestrel Interactive Development System (KIDS) (SmithD9009). On test data for strategic transportation plans provided by U.S. government planners, the generated schedulers (called KTS for Kestrel Transportation Scheduler) solved problems with impressive speed. A typical problem, with 10,000 movement requirements, takes the derived scheduler 1 -- 3 minutes to solve, compared with 2.5 *hours* for a deployed feasibility estimator (JFAST) and 36 *hours* for deployed schedulers (FLOGEN, ADANS). The computed schedules use relatively few resources and satisfy all specified constraints. The speed of this scheduler was due to the synthesis of strong constraint checking and constraint propagation code.

In 1994 Kestrel and BBN began to develop a scheduler to support PACAF (Pacific Air Force) at Hickham AFB, Honolulu which is tasked with in-theater scheduling of a fleet of 26 C-130 cargo aircraft in the Pacific region. We developed (and are continuing to evolve) a domain model of theater transportation scheduling. Several variants of a theater scheduler (called ITAS for In-Theater Airlift Scheduler) have been developed to date, and more are planned. The ITAS interface and was built on top of a commercial database package (Microsoft FoxPro^{tm}) and integrated with the generated LISP-based scheduler by BBN. ITAS runs on an Apple Powerbook laptop computer. The laptop platform makes it attractive both for field and command center operations. ITAS can currently produce ATOs (Air Tasking Orders) based on the schedules that it generates.

The current ITAS scheduler algorithm is another in the KTS family of synthesized algorithms, using a number of new and different constraints from the previous, strategic (inter-theater) model. In this domain, the size of the problems is smaller (small tens of movement requirements, planes and ports), and the time horizon is shorter (typically, less than one week) because the domain is much more reactive. However, there are many more constraints that must be handled, and users need very rapid response from the scheduler in order to be able to iterate on the final schedule, often in less than an hour. On relatively small but realistic problems we have tested so far[6] in ITAS' intended domain of application, the scheduler runs in 5 to 30 seconds on a Macintosh Powerbook 540. Relatively little effort has been spent on optimizing the code to date.

ITAS simultaneously schedules the following classes of resources: (1) aircraft, (2) air crews and their duty day cycles, (3) ground crews for unloading, and (4) ramp space at ports. By its very dynamic nature, the domain presents many special one-time issues that we leave to the user to handle, as discussed later. Basically, ITAS as a system is designed to allow users control the ``shape'' of the final schedule by editing and ``seeding'' the schedule in several ways, and to do daily rescheduling, without losing this user input to the process.

---

[6] Roughly speaking, 10 planes, 10 movement requirements, generating about 50 flights.

In this paper, we describe the current prototype of ITAS. We discuss the constraints from the domain that were handled, and some that have not, as yet, been handled directly in the generated code. We attempt to describe, in informal terms, the algorithm that was produced using KIDS, in an attempt to make clear where this style of scheduling algorithm gets its great speed, and to promote comparison with other constraint-based scheduling systems in the literature. Finally, we briefly describe the nature of user interactions with the scheduler, as they have evolved so far.

AN APPROACH TO SYNTHESIZING SCHEDULERS

Kestrel's approach to developing scheduling software involves several stages. The first step is to develop a formal model of the transportation scheduling domain, called a *domain theory*. Second, the constraints, objectives, and preferences of a particular scheduling problem are stated within a domain theory as a *problem specification.* Finally, an executable scheduler is produced semi-automatically by applying a sequence of *transformations* to the problem specification. The transformations embody programming knowledge about algorithms, data structures, program optimization techniques, etc. The result of the transformation process is executable code that is consistent with the given problem specification. Furthermore, the resulting code can be extremely efficient.

One of the benefits of a transformational approach to scheduling is the synthesis of specialized constraint management code. Previous systems for performing scheduling in AI (e.g. Fox and Smith, 1984; Fox, 1989; Smith.S. et al. 1986, Smith.S. 1989) and Operations Research (Applegate and Cook, 1991; Luenberger, 1989) use constraint representations and operations that are geared for a broad class of problems, such as constraint satisfaction problems or linear programs. In contrast, transformational techniques can derive specialized representations for constraints and related data, and also derive efficient specialized code for constraint checking and constraint propagation.

Our approach tries to make the domain model and scheduling problem explicit and clear, so that, ultimately, users can build new schedulers of the same general class by defining the constraints embodied in their problem and generating a new, situation-specific, scheduler. Basically, the idea is to rapidly develop a situation-specific domain model and problem specification using a knowledge-elicitation system, and then to *synthesize* high-performance planning and scheduling tools that are specialized to the current situation. The majority of users' interaction would be codifying the *domain theory* and specification of the current situation, to aid in synthesizing a customized planning/scheduling tool.

Our current scheduling theories have evolved over months of effort into about 3500 lines of text. It currently takes about 90 minutes to transform our most complex scheduling specification (for ITAS) into optimized and compiled Common Lisp code for Sun workstations. Evolution of the scheduler is performed by evolving the domain theory and specification, followed by regeneration of code. The resulting code, after optimizing transformations, is roughly 3000 lines of code in the REFINE language (depending on how its formatted), which is transformed automatically into about 5200 lines of (largely unreadable) Common LISP code.

CHARACTERIZING THE APPLICATION DOMAIN

A domain theory for scheduling defines the basic concepts of scheduling and the laws for reasoning about the concepts. A scheduling domain model generally consists of models of the *activities* to be scheduled, the *resources* to be used by those activities, *time* (*e.g.*, a time-interval calculus, the *constraints* on the use of resources for activities (capacity constraints), and

the ordering of activities (precedence constraints), and the *utility* of the produced schedule (*e.g.*, minimize a cost objective function).

Using the above concepts we can formulate a variety of scheduling problems. A reservation is a triple consisting of an activity, a resource, and a time interval. Generally, a schedule is a set of *reservations* that satisfy a collection of constraints and optimize (or produce a reasonably good value of) the objective. Transportation scheduling specializes this general notion of scheduling: activities correspond to *movement requirements* and resources correspond to transportation assets such as planes, ships, and trucks. For the ITAS domain, the main assets are planes, but other resources must also be managed including: air crews, ground crews (that load and unload the planes), and parking spaces at airports (for the port constraint called MOG or maximum on ground number).

For ITAS and other transportation schedulers, the activities are called *movement requirements.* ITAS movement requirements include the following information (All times are in seconds from time $\mathbf{t_0}$.):

POE : port	->	*PHIK*
POD : port	->	*PHLI*
ALD : time	->	*0*
EAD : time	->	*86400*
LAD : time	->	*86400*
Loads : integer	->	*20*
Load-type : aircraft-class	->	*C130E*
PAX : integer	->	*1500*
Pallets : integer	->	*50*

Here, the Port of Embarkation (POE) and Port of Debarkation (POD) are the origin and destination of the cargo, respectively. PHIK and PHLI are ICAO codes for Hickham AFB and Lihue Airport, respectively. The ALD (available to load date), EAD (earliest arrival date), LAD (latest arrival date) represent the time bounds on a feasible schedule for the movement requirement. The cargo itself is represented somewhat differently from most strategic transportation models. For a problem at this level (smaller planes, local scheduling), packing of the aircraft is a crucial issue, and irregular size and shape loads must be considered. We do not address the packing problem in this system, so we instead rely on the user to specify irregular cargo in *full Loads* for a specific kind of aircraft, and with a particular configuration of the plane's cargo area (indicated by the Load-type). More regular cargo (PAX and Pallets) may be loaded onto any aircraft that has available space for that type of cargo.

The ITAS schedulers have emphasized efficient search and rich constraint modeling. The current version of ITAS simultaneously schedules the following types of resources, each of which has a variety of constraints associated with it:

1. **Aircraft** are characterized by their capacities, both passenger (PAX) and cargo (pallet) capacities, and travel rate in knots. They also have minimum runway length requirements in order to be acceptable for use at specific airports.

2. **Air crews** typically have 16 hour work days, followed by 12 hours rest. They are not expected to take off until one hour after they are called for duty[7].They also receive a day off

[7] Thus far, we only handle one air crew permanently assigned to each aircraft. This should be fixed by the time of

after a short number of days on duty. When only one crew is available for a plane, the plane does not fly during crew rest periods.

3. **Ground crews** are scheduled for loading and unloading planes at each port.

4. **Parking spaces** at each port are used to model the MOG (Maximum on Ground) constraint on ports where this applies.

5. **Airports** are not treated as resources themselves, but have restrictions like their maximum runway length, and operating hours, and whether they are in a combat zone that impact flights into and out of those locations. They also organize other resources (ground crews, parking spaces).

Ground crews and parking slots (MOG) at airports are essentially aggregate resources in a naive description of the ITAS problem, but are scheduled as if they were individual unit resources in the current system, since KIDS cannot currently generate code that treats these constraints in their aggregate form. This is an area of ongoing research.

ITAS' SCHEDULING CONSTRAINTS

At present, twenty-three constraints characterize a *feasible schedule* for ITAS[8]. The list below collapses the descriptions of some of these together[9]:

1. *Consistent POE and POD* * -- The POE and POD of each movement requirement on a given trip of a resource must be the same as that flight's origin and destination respectively.

2. *Consistent Load Type* * -- Each resource can handle only loads for some movement requirements. For example, a C-141 aircraft can only carry loads of C-141 cargo.

3. *Consistent PAX and Pallet Capacity* * -- For cargo in movement requirements expressed as PAX (passengers) or Pallets (pre-packaged cargo), the capacity of each aircraft to carry that type of cargo cannot be exceeded on a given flight. A flight containing a full *load* can carry no additional PAX or Pallets.

4. *Consistent Release Time* -- The start time of a movement (the flight's earliest departure time (EDT)) must not precede the release time (ALD) of all movement requirements in the flight's manifest, plus the aircraft's load time.

---

this publication.

[8] Some of these constraints (*'d) are not currently given explicitly as part of the top level *Problem Specification* to the KIDS system, but are implicit in the *Global Search Theory*, another part of the domain theory that is essentially an abstract schema describing the branching structure for the search control model to be employed in the generated algorithm. These constraints were ``rolled into" that structure for the sake of the efficiency of the generated code. Future systems will make these constraints explicit again.

[9] The number of corresponding formal constraints is noted in parenthesis. Typically, temporal constraints come in pairs, one that is used to propagate the Earliest Departure Time (EDT) forward, and one that is used to propagate the Latest Departure Time (LDT) backward through the schedule.

5. *Consistent Due Time* -- The finish time of a movement (the flight's latest departure time + flight duration + unload time) must not be later than the Latest Arrival Date (LAD) of all movement requirements in the flight's manifest.

6. *Consistent Flight Separation* -- For flights of the same aircraft, the earliest (latest) departure time plus the flight duration and on-ground time at the destination must be less than the earliest (latest) departure time of the aircraft's next flight.

7. *Consistent Air Crew Usage* -- Only use the given air crews.

8. *Consistent Air Crew Duty Day* -- For a sequence of flights by one crew without a rest period, the earliest (latest) departure time of the last flight, plus the last flight's duration and unload time, should be less than the length of a duty day after the earliest (latest) departure time of the first flight.

9. *Consistent Air Crew Transition* -- By similar formulae, crews get a full 12 hours rest after a duty day plus at least 3.25 hours preparation time before their next flight.

10. *Consistent Air Crew Qualifications* -- Crews must be qualified for the aircraft they are assigned to, the mission types to be flown, and must belong to the unit that `owns' the aircraft. (3 constraints)

11. *Consistent Port Usage* -- Only schedule flights into/out of the given ports.

12. *Consistent Port Runway Length* -- Aircraft cannot land or take off from airports whose maximum runway length is less than the aircraft's minimum take-off/landing runway length[10].

13. *Consistent Port MOG* -- The constraints here are actually on the use of parking spaces, which are assigned individually during search. Basically they state that the earliest (latest) departure time of the aircraft leaving a space must be before the earliest (latest) arrival time of the next aircraft assigned to that space.

14. *Consistent Port Ground Crew* -- Similar to MOG, ground crew reservations must be separated by the corresponding load/unload time. For now, it is assumed that there are a constant number of ground crews available whenever a port is open.

15. *Consistent Port Operating Hours* -- Ports may be closed for take-offs/ landings for some period each day (*e.g.*, night time). (2 constraints)

16. *Consistent Mission Type* -- The cargo in a flight manifest must be from movement requirements with the same mission type (*e.g.*, normal land and unload vs. airdrop).

17. *Consistent Aircraft Usage* -- Only the given aircraft are to be used.

18. *Completeness* -- All movement requirements must be scheduled.

Constraints in the domain theory are defined formally by reference to data structures that are maintained during search, basically mappings of resources (parking spaces, ground crews) to

---

[10]This constraint could be made considerably more complex if it were to take into account whether the plane was loaded or not, its expected fuel level, and the different true minimums for take-off and landing.

sequences of reservations (tuples referencing the aircraft and flight involved). For example, here is a (slightly simplified) constraint on the LAD of a movement requirement:

```
function CONSISTENT-LAD
    (sched : schedule) : boolean
    = forall (ac-indx : integer, flt : integer, mvr : movement-record,
                    flt-indx : integer, lat : time)
        (ac-indx in domain(sched)
            and flt-indx in domain(sched(ac-index).flight-sched)
            and flt = sched(iac-index).flight-sched(flt-indx)
            and mvr in flt.manifest
            and lat = flt.latest-departure-time + flt.flt-duration)
    Implies
        mvr.LAD >= (lat + flt.unload-time)
```

This predicate expresses the constraint that every scheduled movement-record arrives and is unloaded before its latest arrival date[11]. The generated code checks this constraint and propagates any time bounds changes that its enforcement causes.

Similarly, air crew constraints are defined by reference to a mapping from air crews to a data structure containing the sequence of flights they are scheduled to be on, plus other associated information. These data structures are *defined as part of the domain theory*. They are not generated automatically. Data structure design and refinement is an explicit goal of the next generation KIDS-like environment, Specware (Srinivas and Jullig, 1994).

PREFERENTIAL CONSTRAINTS

A typical scheduling problem will involve some choices best expressed as preferences or prioritizations for the use of resources. Although some experiments have been done with KIDS where its search was based in part on a utility or cost function, the current ITAS scheduler does not explicitly handle preferential constraints at all. Instead, at points where choices are made about which resources to use, user-defined functions order the choices. We have experimented with a number of different ordering heuristics, to improve the overall quality of the schedules produced, from the user's perspective.

There are functions (called variously *ASSET-ORDER, AIR-CREW-ORDER* ...) that define the order of consideration of each kind of resource:

- **Movement Requirements** -- *E.g.*, sorted by their LAD, EAD, ALD.
- **Assets (aircraft)** -- *E.g.*, prefer locally available aircraft.
- **Air Crews** -- *E.g.*, prefer the most rested air crews from the aircraft's home unit.
- **Ground Crews** -- Prefer the earliest available crew.
- **Parking Spaces** -- Prefer the earliest available space.

---

[11] A schedule is defined as a sequence of aircraft (which are data structures that contain their flight schedules, also sequences). Thus **sched(i)** indexes the **i**$^{\text{th}}$ aircraft, and **domain(sched)** is the integers **[1..k]** if there are **k** aircraft available.

SYNTHESIZING A SCHEDULER

There are two basic approaches to computing a schedule: local and global. Local methods focus on individual schedules and similarity relationships between them. Once an initial schedule is obtained, it is iteratively improved by moving to neighboring structurally similar schedules. Repair strategies (Zweben et al., 1990; Minton et al, 1990; Biefeld and Cooper, 1990; Selman et al. 1992), and fixed-point iteration (Cai and Paige 1989), and linear programming algorithms are examples of local methods.

Global methods focus on sets of schedules. A feasible or optimal schedule is found by repeatedly splitting an initial set of schedules into subsets until a feasible or optimal schedule can be easily extracted. Backtrack, heuristic search, and branch-and-bound methods are all examples of global methods. We used a global methods to generate the KTS family of schedulers, including ITAS. Other projects taking a global approach include ISIS (Fox and Smith, 1984), OPIS/DITOPS (Smith S., 1989) and MicroBoss (Sadeh, 1991) (all at CMU).

Global search algorithms manipulate sets of candidate solutions. The principal operations are to *extract* candidate solutions from a set and to *split* a set into subsets. Derived operations include various *filters* which are used to eliminate sets containing no feasible or optimal solutions. Starting from an initial set that contains all solutions to the given problem instance, these algorithms repeatedly extracts solutions, splits sets, and eliminates sets via filters until no sets remain to be split. The process is often described as a tree (or DAG) search in which a node represents a set of candidates and an arc represents the split relationship between set and subset. The filters serve to prune off branches of the tree that cannot lead to solutions.

In a simple global search theory of scheduling, schedules are represented as maps from resources to sequences of trips, where each trip includes earliest-start-time, latest-start-time, travel-time, port of embarkation, port of debarkation, and a manifest describing the cargo. This type of schedules has the invariant (or subtype characteristic) that for each trip, the earliest-start-time is no later than the latest-start-time. A partial schedule is a schedule over a subset of the given movement records. Thus the root denotes the set of all candidate solutions found in the tree. This initial (partial) schedule is just the empty schedule -- a map from the available resources to the empty sequence of trips. A partial schedule is extended by first selecting a movement record *mvr* to schedule, then selecting a resource **r**, and then a trip **t** on **r** (either an existing trip or a newly created one) -- the triple **<mvr, r, t>** represents a possible refinement of the current partial schedule in the search space. The alternative ways that a partial schedule can be extended naturally gives rise to the branching structure underlying global search algorithms.

A global search algorithm checks for consistency of the partial solution at each node it explores, pruning those nodes where the test fails. More generally, necessary conditions on the existence of feasible (or optimal) solutions below a node in a branching structure underlie pruning in backtracking and the bounding and dominance tests of branch-and-bound algorithms (Smith D., 1987).

CUTTING CONSTRAINTS AND TEMPORAL CONSTRAINT PROPAGATION

A key technical achievement of the Kestrel work was discovering and implementing technology for generating efficient constraint propagation code. The speed of the KTS schedulers derives from the extremely fast checking and propagation of constraint information at every node of the runtime search tree. Whereas some knowledge-based approaches to scheduling will search a tree at the rate of several nodes per second, some of the synthesized schedulers search *several hundred thousand* nodes per second.

The idea is to derive and utilize the necessary conditions on feasibility of a candidate (partial) schedule. These conditions are called *cutting constraints*. The derived cutting constraints for a particular scheduling problem are analyzed to produce code that iteratively fixes violated constraints (for the current KTS schedulers, by tightening time bounds on reservations) until the cutting constraints are satisfied. This iterative process subsumes the well-known processes of constraint propagation in the AI literature and the notion of cutting planes from the Operations Research literature (Smith D., 1987).

Kestrel researchers developed a general mechanism for deriving constraint propagation code and applied it to scheduling. This model of constraint propagation generalizes the concepts of cutting planes in the Operations Research literature (Nemhauser and Wolsey, 1988) and the forms of propagation studied in the constraint satisfaction literature (e.g. (Hentenryck, 1989)[12])The use of fixed-point iteration for constraint propagation is similar to Paige's work on fixed-point iteration in RAPTS (Cai and Paige, 1989).

KIDS generates propagation code automatically from a subset of the constraints given for a problem, using information in the global search theory schema as a guide. Stephen Westfold of Kestrel was responsible for the design and implementation of the propagation generation subsystem of KIDS (Smith D. and Westfold, 1995). The constraint propagation code that was generated for the original KTS scheduler is nearly as fast as handwritten propagation code for the same problem (cf. Appendix C in (Smith D. 1992)). The propagation code for the current ITAS is many times more complicated, and it would have been quite difficult to generate it by hand.

The generated propagation code called at each node during search operates much the way a temporal constraint system does, although the code is targeted to the problem domain using specific knowledge of the particular constraints handled, the choices being considered at that point by the scheduler, and a logical analysis of the possible effects of tightening particular time bounds. The result is code tailored to the specific scheduling problem addressed, based on the constraints specified in the domain theory. The overall effect is very efficient pruning of the search space. At each node in the space, after each potential choice is made by satisfying all relevant non-temporal necessary constraints, the propagation code is run to see if the schedule is still viable. If so, the system recurs on the new refined state. Otherwise, a new choice is made.

THE ITAS SCHEDULER ALGORITHM

This section provides a brief outline of the generated scheduler's algorithm, in order to convey a sense of how the system works, and (hopefully) why it is fast. Due to space limitations, we have greatly simplified and summarized what goes on. The code itself is extremely complex and barely readable, with few function breaks and numerous generated intermediate variables.

In ITAS, a *schedule* is a list of all of the aircraft, where each aircraft is an object that contains its (partial) schedule. A *sortie* is essentially a short sequence of flights that contains (at most) a positioning flight to get to a POE, a POE-to-POD flight, and a recovery flight away from the POD to a refueling or resting station.

The top level function, **KTS**, just calls *KTS-AUX*, which recursively searches the space of partial schedules. **KTS** then tests to see if the result is defined and *UNSCHED-MVRS* is empty, in which case it extracts a solution from the current search state.

---

[12] For details of deriving pruning mechanisms for other problems see (Smith D. 1987, Smith D. 1990, Smith D. and Lowry, 1990, Smith D. 1991).

The arguments to *KTS-AUX* constitute a state in the search space:

1. *MVRS:* a sorted list of all original movement requirements
2. *ASSETS:* a list of all available aircraft for the problem
3. *PORTS:* a list of all of the ports in the problem
4. *AIR-CREWS:* a list of all air crews
5. *PSCHED:* a partially completed schedule
6. *SCHED-MVRS:* all of the movement requirements that are already scheduled
7. *UNSCHED-MVRS:* all of the remaining unscheduled movement requirements
8. *AIR-CREW-MAP:* a mapping of air crews to their schedules
9. *GND-CREW-MAP:* a mapping of ground crews to their schedules
10. *GS-FLTS:* flights remaining to be scheduled for the current sortie
11. *PRK-SLOT-MAP:* a mapping of parking spots at ports to their reservations

KTS-AUX for the ITAS problem splits the search into four main branches, as defined by the global search theory schema of the domain theory. The first two branches consider refinements of the current partial schedule for flights that are part of the current sortie. A current sortie exists while a movement requirement is being processed, but not all of the necessary flights (positioning, cargo-moving, depositioning) have been scheduled (so *GS-FLTS* is not empty). The third and fourth branches consider the next movement requirement, and search to place as much of that requirement as possible on an existing or new flight, respectively. In the process, a new sortie is created.

Within each of the four main branches of *KTS-AUX*, one of two versions of *SPLIT&PROPAGATE* is called, to generate the proposed new state, and run the propagation routines for each temporal constraint that might be violated at that point. If *SPLIT&PROPAGATE* succeeds and returns a new state, When *KTS-AUX* finds a possible a new state, it calls *SPLIT&PROPAGATE* with these additional arguments *S-State* to represent the currently considered resources:

1. *FLT:* the currently considered flight, if any
2. *POE-TO-POD-FLT:* the load-carrying flight for this move reqt., if known
3. *AIR-CREW:* the flight's air crew
4. *CURRENT-DUTY-DAY?:* Boolean indicating to search in the current (next) duty day
5. *GROUND-CREW:* the ground crew scheduled to unload the plane, if defined
6. *GND-CREW-RES:* the reservation for that crew, if defined.
7. *PARKING-SLOT:* the parking slot to be used at the destination, if defined

The branches, considered sequentially, are:

1. If *GS-FLTS* is not empty, consider all possible ground crews and parking slots for
   FLT = first(GS-FLTS)

and

MVR = first(UNSCHED-MVRS),

by calling *SPLIT&PROPAGATE1* on each, but considering *only flights in the current duty day* of the flight's air crew.

2. If *GS-FLTS* is not empty, consider all possible ground crews and parking slots for

   FLT = first(GS-FLTS

   and

   MVR = first(UNSCHED-MVRS),

   by calling *SPLIT&PROPAGATE1* on each, but considering *only flights in the next duty day* of the flight's air crew.

3. If *GS-FLTS* is empty, let

   MVR = first(UNSCHED-MVRS),

   and consider all cargo-carrying flights of aircraft in the current *S-State*'s partial schedule that have remaining capacity for *MVR* by calling *SPLIT&PROPAGATE2* on each.

4. If *GS-FLTS* is empty, let

   MVR = first(UNSCHED-MVRS),

   and consider all aircraft in *ASSET-ORDER*, calling *SPLIT&PROPAGATE2* on each, which creates a flight sortie for that *MVR*, and tests that requirement's LAD can be met.

5. (optional) If no feasible (partial) schedule is found, relax the LAD of the last movement requirement.

Each *SPLIT&PROPAGATE* function contains a version of the generated constraint checking and propagation code for a different set of conditions, depending on how the partial schedule has been changed. Essentially, these functions take the constraints stated in the problem description, in turn, and, if they are temporal constraints, propagate their bounds either forward or backward in time, depending on whether the earliest (EDT) or latest departure time (LDT) of a flight is referenced in the constraint. For example, *SPLIT&PROPAGATE1*, which is used to schedule an existing flight in *GS-FLTS*, does the following:

1. Check *CONSISTENT-RUNWAY-LENGTH.*
2. Check *CONSISTENT-MISSION-TYPE.*
3. Generate a state vector with reservations for the Ground Crew, Air Crew, and Parking Slot assignments for the flight
4. Propagate EDT, checking *CONSISTENT-AIR-CREW-DUTY-DAY-EDT*
5. Propagate LDT, checking *CONSISTENT-AIR-CREW-DUTY-DAY-LDT*
6. Propagate EDT, checking *CONSISTENT-FLIGHT-SEPARATION-EDT* for all aircraft used in *PSCHED.*
7. Propagate LDT, checking *CONSISTENT-FLIGHT-SEPARATION-LDT* for all aircraft used in *PSCHED.*
8. Propagate EDT, checking *CONSISTENT-ALD* for all *MVR* in the manifest.
9. Propagate EDT, checking *CONSISTENT-GROUND-UNLOAD-CREW-EDT* for all of the current ground crews at the destination.

10. Propagate LDT, checking *CONSISTENT-GROUND-UNLOAD-CREW-LDT* for all of the current ground crews at the destination.

Each propagation of the EDT or LDT above is based on recursively checking the corresponding constraints for *CONSISTENT-FLIGHT-SEPARATION, CONSISTENT-AIR-CREW-DUTY-DAY, CONSISTENT-AIR-CREW-TRANSITION,* and *CONSISTENT-MOG.*

If the state is still defined (consistent) at the end of all of this, the proposed state is accepted, the movement requirement is marked scheduled and *KTS-AUX* is called recursively.

SPLIT&PROPAGATE2 is similar, but somewhat more complicated, since it generates a sortie of flights for the next movement requirement, and finds an aircraft and crew for that sortie.

CONSTRAINT RELAXATION

Many scheduling problems are over-constrained. Over-constrained problems are typically handled by relaxing the constraints. The usual method, known as Lagrangian Relaxation (Nemhauser and Wolsey, 1988), is to move constraints into the objective function. This entails reformulating the constraint so that it yields a quantitative measure of how well it has been satisfied.

Our current experimental approach is to relax the input data just enough that a feasible solution exists (the optional branch 5 of *KTS-AUX*). This approach is available as an option in ITAS to avoid an exhaustive search when no feasible solution may exist. This mode relaxes the LAD (Latest Arrival Date) constraint, when the current search reaches an impasse, rather than backing up. The relaxation takes place only when there is no feasible solution to the problem data. ITAS uses the difference between the arrival date of a trip and the LAD of a movement requirement in that trip as a measure of how much to relax the LAD. The relaxation is such as to minimally delay the arrival of the requirement to its POD.

SYSTEM DESIGN AND USER INTERACTIONS ISSUES WITH ITAS

The target user community for ITAS needed a tool that was fast, easy to use, flexible and portable. These factors motivated us to develop the system on Macintoshes, so that it could be deployed on a Mac Powerbook notebook computer. Given the speed of the scheduler, size of the problems to be faced, and the short scheduling horizon, we expect that most problems will run in under a minute with the current system. We used Microsoft FoxPro[tm] as the primary database and interface substrate, because it runs on Macintoshes and it was simple to understand. For the current prototype, data exchanges with the scheduler (running in Macintosh Commonlisp) are done using a simple combination of Applescript and file I/O. On our current suite of test problems, it takes as long to do the inter-module I/O as it does to build a schedule.

The interface uses Apple's menu system and set of data entry screens, organized around two types of data: *background data* (for caches of locations (airports), units, aircraft types and characteristics, frequently used aircraft, etc.) and *situation data*, the current collection of aircraft, crews, ports and movement requirements for which a schedule is being built. The primary graphical schedule display is a mouse sensitive version of a schedule Gantt chart modeled after what the airlift schedulers call their `Rainbow' chart, because the different types of aircraft are assigned different colors.

There is also a flight editor that enables users to make modifications to the existing schedule before exporting it, and to iterate on a solution with the scheduler in the loop. Users of this tool need to be able to edit the schedules produced, when circumstances not captured by the available

constraints come up. We needed to make it possible for parts of the schedule to be specified directly by the user, and not revised by the scheduler.

The ITAS domain is extremely reactive. The people who now generate schedules by hand do so a day at a time, since their information is often only good enough to *estimate* what will be required three, or even two days hence. Given a tool that will build schedules for them, they will likely extend their planning horizon (currently 10 days or less), at least for plan/resource evaluation purposes, but for day to day operations, they will continue to schedule only a few days at a time, and do daily rescheduling, as needed.

In support of rescheduling, users can mark (or edit then mark) flights in the existing current schedule as `Frozen'. ITAS-KTS takes as part of its input the leading frozen edge of a schedule, and uses it only to properly reflect resource availability, and so that those flights and their itineraries are reflected in its output. Because the ITAS-KTS scheduling algorithm generates candidate flights in a time forward fashion, a frozen flight necessarily means that all prior flights by that aircraft are also frozen. Thus the current scheduler does not give users as much flexibility to edit and then rerun schedules as is found in some other global-search based rescheduling systems whose search is not time ordered (*e.g.*, (Smith S., 1989)) or in systems using local repair strategies (Zweben et al. 1990). We are planning to generate an ITAS scheduler that inserts flights in a schedule rather than adding them to the end of schedule. The interesting issue is to evaluate the resulting tradeoff between the slowdown due to greatly increased search tree branching and the extra flexibility gained during (re)scheduling.

Our users also wished to be able to lay out patterns of flight itineraries for some sets of aircraft, based on their paper process. For example, it is customary for them, when scheduling a group of planes carrying loads to the same destination, to stagger the departure times to assure a uniform flow into an airport. We have not yet been able to reproduce this style of output automatically, but have provided a mechanism that allows users to lay out itineraries (sequences of ports to visit) for sets of aircraft, each departure offset by some specified time interval. Flights of this kind that are seen by ITAS-KTS as having fixed earliest departure times, but infinite latest departure times, so they can effectively be ``slid forward" in time to accommodate port and ground crew constraints. A planned extension of this capability is to provide these ``flows" to the scheduler as patterns that can be used repeatedly in normal scheduling, especially where the distance is great enough that an intermediate stop en route is required when transporting some cargo. The current scheduler does not support multi-leg trips between cargo sources and destinations.

CONCLUSIONS

This project was conceived of as a short term effort to demonstrate the potential and practical utility of automatically generated scheduling software. Given the previously demonstrated speed of the algorithms produced and this demonstration that the technique can be scaled up to realistic sets of constraints, we believe that those conclusions are clearly justified. This paper is an attempt to make clear how the current class of generated schedulers work, why they are fast, and what some of the tradeoffs were in terms of flexibility. We believe that it will often be better to generate, and make available through a single interface, a suite of quick schedulers, tailored to different constraints, than to have one scheduler that is slow on all problems.

The great advantage of the synthesis approach is the ability to expose problem structure and exploit it by transformationally deriving efficient problem-specific code. In this case, the reuse of design knowledge (global search, constraint propagation) made an enormous difference in performance. Future directions include making it easier for domain experts to specify their own

constraints, and generate their own problem-specific schedulers, and broadening the classes of scheduling algorithms that can be generated, and the types of constraints that can be handled. The next generation of KIDS, Specware, will also address the need for more automatic and controllable data structure design. We should note that a preliminary test indicates that there is at least *an order of magnitude speedup* still to be had, since the current scheduler uses *lists* for all of its object data structures! Specware will be able to use a variety of data types, and generate code in other languages than LISP (*e.g.*, C++).

On the interface side, we seek to exploit the great speed of even the current class of schedulers by greatly increasing the interactive nature of the scheduling task, providing more automatic capabilities in the area of resource analysis and comparative schedule analysis.

ACKNOWLEDGMENTS

The authors would like to thank the following people for their important contributions to the ITAS effort: Don Roberts and John Lemmer of Rome Laboratory; Stephen Westfold of Kestrel Institute; and LtCol Chris Stuhldreher, Maj Keith LaCrosse, and Cpt Rob Burgess of the PACAF Airlift Operations Center.

BIBLIOGRAPHY

Applegate, D., and Cook, W. 1991. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing* 3(2):149-156.

Biefeld, E., and Cooper, L. 1990. Operations mission planner. Technical Report JPL 90-16, Jet Propulsion Laboratory.

Cai, J., and Paige, R. 1989. Program derivation by fixed point computation. *Science of Computer Programming* 11:197-261.

Fox, M. S., and Smith, S. F. 1984. ISIS -- a knowledge-based system for factory scheduling. *Expert Systems* 1(1):25--49.

Fox, M.S., Sadeh, N., and Baykan, C. 1989. Constrained heuristic search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* 309--315.

Hentenryck, P. V. 1989. *Constraint Satisfaction in Logic Programming.* Cambridge, MA: Massachusetts Institute of Technology.

Luenberger, D. G. 1989. *Linear and Nonlinear Programming.* Reading, MA: Addison-Wesley Publishing Company, Inc.

Minton, S., Johnson, M., Philips, A.B., and Laird, P. 1990. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the Eighth National Conference on Artificial Intelligence.,* 290--295.

Nemhauser, G. L., and Wolsey, L. A. 1988. *Integer and Combinatorial Optimization.* New York: John Wiley & Sons, Inc.

Sadeh, N. 1991. Look-ahead techniques for micro-opportunistic job shop scheduling. Technical Report CMU-CS-91-102, Carnegie-Mellon University.

Selman, B., Levesque, H. and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence,* 440-446.

Smith, D. R. 1987. Structure and design of global search algorithms. Technical Report KES.U.87.12, Kestrel Institute.

Smith, D. R. September 1990. KIDS -- a semi-automatic program development system. *IEEE Transactions on Software Engineering Special Issue on Formal Methods in Software Engineering* 16(9):1024--1043.

Smith, D. R., and Lowry, M. R. 1990. Algorithm theories and design tactics. *Science of Computer Programming* 14(2-3):305-321.

Smith, D. R. 1991. KIDS: A knowledge-based software development system. In Lowry, M., and McCartney, R., eds., *Automating Software Design.* Menlo Park: MIT Press. 483-514.

Smith, D. R. 1992. Transformational approach to scheduling. Technical Report KES.U.92.2, Kestrel Institute.

Smith, D. R., and Westfold, S. J. 1995. Synthesis of constraint algorithms. In Saraswat, V., and Hentenryck, P. V., eds., *Principles and Practice of Constraint Programming.* Cambridge, MA: The MIT Press.

Smith, D. R., Parra, E. A., and Westfold, S. J. 1995. Synthesis of high performance transportation schedulers. Technical Report KES.U.95.6, Kestrel Institute.

Smith, S. F., Fox, M. S, and Ow, P.S. 1986. Constructing and maintaining detailed production plans: Investigations into the development of knowledge-based factory scheduling systems. *AI Magazine* 7(4):45-61.

Smith, S. F. 1989. The OPIS framework for modeling manufacturing systems. Technical Report CMU-RI-TR-89-30, The Robotics Institute, Carnegie Mellon University.

Srivivas, Y. V., and Jullig, R. 1994. Specware:[tm] formal support for composing software. Technical Report KES.U.94.5, Kestrel Institute. In *Proceedings of the Conference on Mathematics of Program Construction*, Kloster Irsee, Germany, July, 1995.

Zweben, M., Deal, M., and Gargan, R. 1990. Anytime rescheduling. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control,* 215-219. San Diego, CA: DARPA.

DISTRIBUTION LIST

addresses	number of copies
DONALD ROBERTS RL/C3CA 525 BROOKS RD ROME NY 13441-4505	10
DR MARK H BURSTEIN BBN CORP 10 MOULTON ST CAMBRIDGE MA 02138	5
ROME LABORATORY/SUL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
RELIABILITY ANALYSIS CENTER 201 MILL ST. ROME NY 13440-8200	1
ROME LABORATORY/C3AB 525 BROOKS RD ROME NY 13441-4505	1
ATTN: RAYMOND TADROS GIDEP P.O. BOX 8000 CORONA CA 91718-8000	1

AFIT ACADEMIC LIBRARY/LDEE 1
2950 P STREET
AREA B, BLDG 642
WRIGHT-PATTERSON AFB OH 45433-7765

ATTN: R.L. DENISON 1
WRIGHT LABORATORY/MLPO, BLDG. 651
3005 P STREET, STE 6
WRIGHT-PATTERSON AFB OH 45433-7707

ATTN: GILBERT G. KUPERMAN 1
AL/CFHI, BLDG. 248
2255 H STREET
WRIGHT-PATTERSON AFB OH 45433-7022

OL AL HSC/HRG, BLDG. 190 1
2698 G STREET
WRIGHT-PATTERSON AFB OH 45433-7604

AUL/LSAD 1
600 CHENNAULT CIRCLE, BLDG. 1405
MAXWELL AFB AL 36112-6424

US ARMY STRATEGIC DEFENSE COMMAND 1
CSSD-IM-PA
P.O. BOX 1500
HUNTSVILLE AL 35807-3801

COMMANDING OFFICER 1
NCCOSC RDT&E DIVISION
ATTN: TECHNICAL LIBRARY,CODE 0274
53560 HULL STREET
SAN DIEGO CA 92152-5001

COMMANDER, TECHNICAL LIBRARY 1
4747000/C0223
NAVAIRWARCENWPNDIV
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6001

SPACE & NAVAL WARFARE SYSTEMS 2
COMMAND (PMW 178-1)
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200

SPACE & NAVAL WARFARE SYSTEMS 1
COMMAND, EXECUTIVE DIRECTOR (PD13A)
ATTN: MR. CARL ANDRIANI
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200

COMMANDER, SPACE & NAVAL WARFARE 1
SYSTEMS COMMAND (CODE 32)
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200

CDR, US ARMY MISSILE COMMAND 2
RSIC, BLDG. 4484
AMSMI-RD-CS-R, DOCS
REDSTONE ARSENAL AL 35898-5241

ADVISORY GROUP ON ELECTRON DEVICES 1
SUITE 500
1745 JEFFERSON DAVIS HIGHWAY
ARLINGTON VA 22202

REPORT COLLECTION, CIC-14 1
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545

AEDC LIBRARY 1
TECHNICAL REPORTS FILE
100 KINDEL DRIVE, SUITE C211
ARNOLD AFB TN 37389-3211

COMMANDER 1
USAISC
ASHC-IMD-L, BLDG 61801
FT HUACHUCA AZ 85613-5000

US DEPT OF TRANSPORTATION LIBRARY
FB10A, M-457, RM 930
800 INDEPENDENCE AVE, SW
WASH DC 22591 — 1

AIR WEATHER SERVICE TECHNICAL
LIBRARY (FL 4414)
859 BUCHANAN STREET
SCOTT AFB IL 62225-5118 — 1

AFIWC/MSO
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016 — 1

SOFTWARE ENGINEERING INSTITUTE
CARNEGIE MELLON UNIVERSITY
4500 FIFTH AVENUE
PITTSBURGH PA 15213 — 1

NSA/CSS
K1
FT MEADE MD 20755-6000 — 1

DCMAD/WICHITA/GKEP
SUITE B-34
401 N MARKET STREET
WICHITA KS 67202-2095 — 1

PHILLIPS LABORATORY
PL/TL (LIBRARY)
5 WRIGHT STREET
HANSCOM AFB MA 01731-3004 — 1

THE MITRE CORPORATION
ATTN: E. LADURE
D460
202 BURLINGTON RD
BEDFORD MA 01732 — 1

OUSD(P)/DTSA/DUTD
ATTN: PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202 — 2

DR JAMES ALLEN
COMPUTER SCIENCE DEPT/BLDG RM 732
UNIV OF ROCHESTER
WILSON BLVD
ROCHESTER NY 14627 1

DR YIGAL ARENS
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 1

DR RAY BAREISS
THE INST. FOR LEARNING SCIENCES
NORTHWESTERN UNIV
1890 MAPLE AVE
EVANSTON IL 60201 1

DR MARIE A. BIENKOWSKI
SRI INTERNATIONAL
333 RAVENSWOOD AVE/EK 337
MENLO PRK CA 94025 1

DR PIERO P. BONISSONE
GE CORPORATE RESEARCH & DEVELOPMENT
BLDG K1-RM 5C-32A
P. O. BOX 8
SCHENECTADY NY 12301 1

MR. DAVID BROWN
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269 1

DR MARK BURSTEIN
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138 1

DR GREGG COLLINS
INST FOR LEARNING SCIENCES
1890 MAPLE AVE
EVANSTON IL 60201 1

DR STEPHEN E. CROSS
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213 1

DR THOMAS CHEATHAM 1
HARVARD UNIVERSITY
DIV OF APPLIED SCIENCE
AIKEN, RM 104
CAMBRIDGE MA 02138

MS. LAURA DAVIS 1
CODE 5510
NAVY CTR FOR APPLIED RES IN AI
NAVAL RESEARCH LABORATORY
WASH DC 20375-5337

DR THOMAS L. DEAN 1
BROWN UNIVERSITY
DEPT OF COMPUTER SCIENCE
P.O. BOX 1910
PROVIDENCE RI 02912

DR WESLEY CHU 1
COMPUTER SCIENCE DEPT
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

DR PAUL R. COHEN 1
UNIV OF MASSACHUSETTS
COINS DEPT
LEDERLE GRC
AMHERST MA 01003

DR JON DOYLE 1
LABORATORY FOR COMPUTER SCIENCE
MASS INSTITUTE OF TECHNOLOGY
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139

DR. BRIAN DRABBLE 1
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH/80 S. BRIDGE
EDINBURGH EH1 1HN
UNITED KINGDOM

MR. SCOTT FOUSE 1
ISX CORPORATION
4353 PARK TERRACE DRIVE
WESTLAKE VILLAGE CA 91361

MR. STU DRAPER 1
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269

DR MARK FOX
DEPT OF INDUSTRIAL ENG
UNIV OF TORONTO
4 TADDLE CREEK ROAD
TORONTO, ONTARIO, CANADA
1

MR. GARY EDWARDS
ISX CORPORATION
2000 N 15TH ST, SUITE 1000
ARLINGTON, VA 22201
1

MR. RUSS FREW
GENERAL ELECTRIC
MOORESTOWN CORPORATE CENTER
BLDG ATK 145-2
MOORESTOWN NJ 08057
1

DR MICHAEL FEHLING
STANFORD UNIVERSITY
ENGINEERING ECO SYSTEMS
STANFORD CA 94305
1

RICK HAYES-ROTH
CIMFLEX-TEKNOWLEDGE
1810 EMBARCADERO RD
PALO ALTO CA 94303
1

DR JIM HENDLER
UNIV OF MARYLAND
DEPT OF COMPUTER SCIENCE
COLLEGE PARK MD 20742
1

MS. YOLANDA GIL
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292
1

MR. MORTON A. HIRSCHBERG, DIRECTOR
US ARMY RESEARCH LABORATORY
ATTN: AMSRL-CI-CB
ABERDEEN PROVING GROUND MD
21005-5066
1

MR. MARK A. HOFFMAN
ISX CORPORATION
1165 NORTHCHASE PARKWAY
MARIETTA GA 30067
1

DR RON LARSEN 1
NAVAL CMD, CONTROL & OCEAN SUR CTR
RESEARCH, DEVELOP, TEST & EVAL DIV
CODE 444
SAN DIEGO CA 92152-5000

DR CRAIG KNOBLOCK 1
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

MR. RICHARD LOWE (AP-10) 1
SRA CORPORATION
2000 15TH STREET NORTH
ARLINGTON VA 22201

MR. TED C. KRAL 1
BBN SYSTEMS & TECHNOLOGIES
4015 HANCOCK STREET, SUITE 101
SAN DIEGO CA 92110

DR JOHN LOWRENCE 1
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PARK CA 94025

DR. ALAN MEYROWITZ 1
NAVAL RESEARCH LABORATORY/CODE 5510
4555 OVERLOOK AVE
WASH DC 20375

ALICE MULVEHILL 1
BBN
10 MOULTON STREET
CAMBRIDGE MA 02238

DR ROBERT MACGREGOR 1
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL REY CA 90292

DR DREW MCDERMOTT 1
YALE COMPUTER SCIENCE DEPT
P.O. BOX 2158, YALE STATION
51 PROPSPECT STREET
MEW HAVEN CT 06520

DR DOUGLAS SMITH 1
KESTREL INSTITUTE
3260 HILLVIEW AVE
PALO ALTO CA 94304

DR. AUSTIN TATE 1
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH
80 SOUTH BRIDGE
EDINBURGH EH1 1HN - SCOTLAND

DIRECTOR 1
DARPA/ITO
3701 N. FAIRFAX DR., 7TH FL
ARLINGTON VA 22209-1714

DR STEPHEN F. SMITH 1
ROBOTICS INSTITUTE/CMU
SCHENLEY PRK
PITTSBURGH PA 15213

DR. ABRAHAM WAKSMAN 1
AFOSR/NM
110 DUNCAN AVE., SUITE B115
BOLLING AFB DC 20331-0001

DR JONATHAN P. STILLMAN 1
GENERAL ELECTRIC CRD
1 RIVER RD, RM K1-5C31A
P. O. BOX 8
SCHENECTADY NY 12345

DR EDWARD C.T. WALKER 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

DR BILL SWARTOUT 1
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

GIO WIEDERHOLD 1
STANFORD UNIVERSITY
DEPT OF COMPUTER SCIENCE
438 MARGARET JACKS HALL
STANFORD CA 94305-2140

DR KATIA SYCARA/THE ROBOTICS INST 1
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIV
DOHERTY HALL RM 3325
PITTSBURGH PA 15213

DR DAVID E. WILKINS 1
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PARK CA 94025

DR. PATRICK WINSTON 1
MASS INSTITUTE OF TECHNOLOGY
RM NE43-817
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139

DR JOHN P. SCHILL 1
ARPA/ISO
3701 N FAIRFAX DRIVE
ARLINGTON VA 22203-1714

DR STEVE ROTH 1
CENTER FOR INTEGRATED MANUFACTURING
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIV
PITTSBURGH PA 15213-3890

DR YOAV SHOHAM 1
STANFORD UNIVERSITY
COMPUTER SCIENCE DEPT
STANFORD CA 94305

MR. MIKE ROUSE 1
AFSC
7800 HAMPTON RD
NORFOLK VA 23511-6097

DR LARRY BIRNBAUM 1
NORTHWESTERN UNIVERSITY
ILS
1890 MAPLE AVE
EVANSTON IL 60201

MR. LEE ERMAN 1
CIMFLEX TECKNOWLEDGE
1810 EMBARCARDERO RD
PALO ALTO CA 94303

DR MATTHEW L. GINSBERG 5
CIRL, 1269
UNIVERSITY OF OREGON
EUGENE OR 97403

MR IRA GOLDSTEIN 1
OPEN SW FOUNDATION RESEARCH INST
ONE CAMBRIDGE CENTER
CAMBRIDGE MA 02142

MR JEFF GROSSMAN, CO 1
NCCOSC RDTE DIV 44
5370 SILVERGATE AVE, ROOM 1405
SAN DIEGO CA 92152-5146

JAN GUNTHER 1
ASCENT TECHNOLOGY, INC.
64 SIDNEY ST, SUITE 380
CAMBRIDGE MA 02139

DR LYNETTE HIRSCHMAN 1
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730

DR ADELE E. HOWE 1
COMPUTER SCIENCE DEPT
COLORADO STATE UNIVERSITY
FORT COLLINS CO 80523

DR LESLIE PACK KAELBLING 1
COMPUTER SCIENCE DEPT
BROWN UNIVERSITY
PROVIDENCE RI 02912

DR SUBBARAO KAMBHAMPATI 1
DEPT OF COMPUTER SCIENCE
ARIZONA STATE UNIVERSITY
TEMPE AZ 85287-5406

MR THOMAS E. KAZMIERCZAK 1
SRA CORPORATION
331 SALEM PLACE, SUITE 200
FAIRVIEW HEIGHTS IL 62208

DR CARLA GOMES 1
ROME LABORATORY/C3CA
525 BROOKS RD
ROME NY 13441-4505

DR MARK T. MAYBURY 1
ASSOCIATE DIRECTOR OF AI CENTER
ADVANCED INFO SYSTEMS TECH G041
MITRE CORP, BURLINGTON RD, MS K-329
BEDFORD MA 01730

MR DONALD P. MCKAY 1
PARAMAX/UNISYS
P O BOX 517
PAOLI PA 19301

DR KAREN MYERS 1
AI CENTER
SRI INTERNTIONAL
333 RAVENSWOOD
MENLO PARK CA 94025

DR MARTHA E POLLACK 1
DEPT OF COMPUTER SCIENCE
UNIVERSITY OF PITTSBURGH
PITTSBURGH PA 15260

DR RAJ REDDY 1
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213

DR EDWINA RISSLAND 1
DEPT OF COMPUTER & INFO SCIENCE
UNIVERSITY OF MASSACHUSETTS
AMHERST MA 01003

DR MANUELA VELOSO 1
CARNEGIE MELLON UNIVERSITY
SCHOOL OF COMPUTER SCIENCE
PITTSBURGH PA 15213-3891

DR DAN WELD 1
DEPT OF COMPUTER SCIENCE & ENG
MAIL STOP FR-35
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195

MR JOE ROBERTS 1
ISX CORPORATION
4301 N FAIRFAX DRIVE, SUITE 301
ARLINGTON VA 22203

DR TOM GARVEY 1
ARPA/ISO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

MR JOHN N. ENTZMINGER, JR. 1
ARPA/DIRO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

DIRECTOR 1
ARPA/ISO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

OFFICE OF THE CHIEF OF NAVAL RSCH 1
ATTN: MR PAUL QUINN
CODE 311
800 N. QUINCY STREET
ARLINGTON VA 22217

DR OREN ETZIONI 1
DEPT OF COMPUTER SCIENCE
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195

DR GEORGE FERGUSON 1
UNIVERSITY OF ROCHESTER
COMPUTER STUDIES BLDG, RM 732
WILSON BLVD
ROCHESTER NY 14627

DR STEVE HANKS 1
DEPT OF COMPUTER SCIENCE & ENG'G
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195

MR DON MORROW 1
BBN SYSTEMS & TECHOLOGIES
101 MOONGLOW DR
BELLEVILLE IL 62221

DR CHRISTOPHER OWENS
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON ST
CAMBRIDGE MA 02138
1

DR ADNAN DARWICHE
INFORMATION & DECISION SCIENCES
ROCKWELL INT'L SCIENCE CENTER
1049 CAMINO DOS RIOS
THOUSAND OAKS CA 91360
1

DR JAIME CARBONNEL
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIVERSITY
DOHERTY HALL, ROOM 3325
PITTSBURGH PA 15213
1

DR NORMAN SADEH
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIVERSITY
DOHERTY HALL, ROOM 3315
PITTSBURGH PA 15213
1

DR JAMES CRAWFORD
CIRL, 1269
UNIVERSITY OF OREGON
EUGENE OR 97403
1

DR TAIEB ZNATI
UNIVERSITY OF PITTSBURGH
DEPT OF COMPUTER SCIENCE
PITTSBURGH PA 15260
1

DR MARIE DEJARDINS
SRI INTERNATIONAL
333 RAVENSWOOD AVENUE
MENLO PARK CA 94025
1

ROBERT J. KRUCHTEN
HQ AMC/SCA
203 W LOSEY ST, SUITE 1016
SCOTT AFB IL 62225-5223
1

DR. DAVE GUNNING
DARPA/ISO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714
1

MS. LEAH WONG
NCCOSC RDTE DIVISIION
53560 HULL STREET
SAN DIEGO CA 92152-5001 1

B. MCMURREY
NCCOSC RDT&E DIVISION
CODE 421
53560 HULL STREET
SAN DIEGO CA 95152-5001 1

GINNY ALBERT
LOGICON ITG
2100 WASHINGTON BLVD
ARLINGTON VA 22204 1

DAVID HESS
SAIC ATLANTIC PROGRAMS
ONE ENTERPRISE PARKWAY, SUITE 370
HAMPTON VA 23666 1

COL ROBERT PLEBANEK
DARPA/ISO
3701 N FAIRFAX DR
ARLINGTON VA 22203 1

ADAM PEASE
TECKNOWLEDGE
1810 EMBARCADERO RD
PALO ALTO CA 94303 1

JIM SHOOP
ISX CORPORATION
4353 PARK TERRACE DR
WESTLAKE VILLAGE CA 91361 1

DR ROBERT NECHES
DARPA/ISO
3701 N FAIRFAX DR
ARLINGTON VA 22203 1

DAVID SWANSON
NRAD
53118 GATCHELL RD
44207 BLDG 600 RM 422C
SAN DIEGO CA 92152 1

DR STEPHEN WESTFOLD 1
KESTREL INSTITUTE
3260 HILLVIEW AVE
PALO ALTO CA 94304

MAJ TOMMY LANCASTER 1
USTRANSCOM/TCJ5-SC
508 SCOTT DR
SCOTT AFB IL 62225-5357

JAMES APPLEGATE 1
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269

SOFTWARE ENGINEERING INSTITUTE 1
CARNEGIE MELLON UNIVERSITY
4500 FIFTH AVE
PITTSBURGH PA 15213

DIRNSA 1
R509
9800 SAVAGE RD
FT MEADE MD 20755-6000

NSA/CSS 1
K1
FT MEADE MD 20755-6000

DCMAO/WICHITA/GKEP 1
SUITE B-34
401 N MARKET ST
WICHITA KS 67202-2095

PHILLIPS LABORATORY 1
PL/TL (LIBRARY)
5 WRIGHT STREET
HANSCOM AFB MA 01731-3004

THE MITRE CORPORATION 1
ATTN: E LADURE
D460
202 BURLINGTON RD
BEDFORD MA 01732

OUSD (P)/DTSA/DUTD
ATTN: PATRICK G. SULLIVAN, JR
400 ARMY NAVY DR
SUITE 300
ARLINGTON VA 22202

1

*U.S. GOVERNMENT PRINTING OFFICE: 1997-509-127-61005

MISSION OF ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Material Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.